

MacTech®

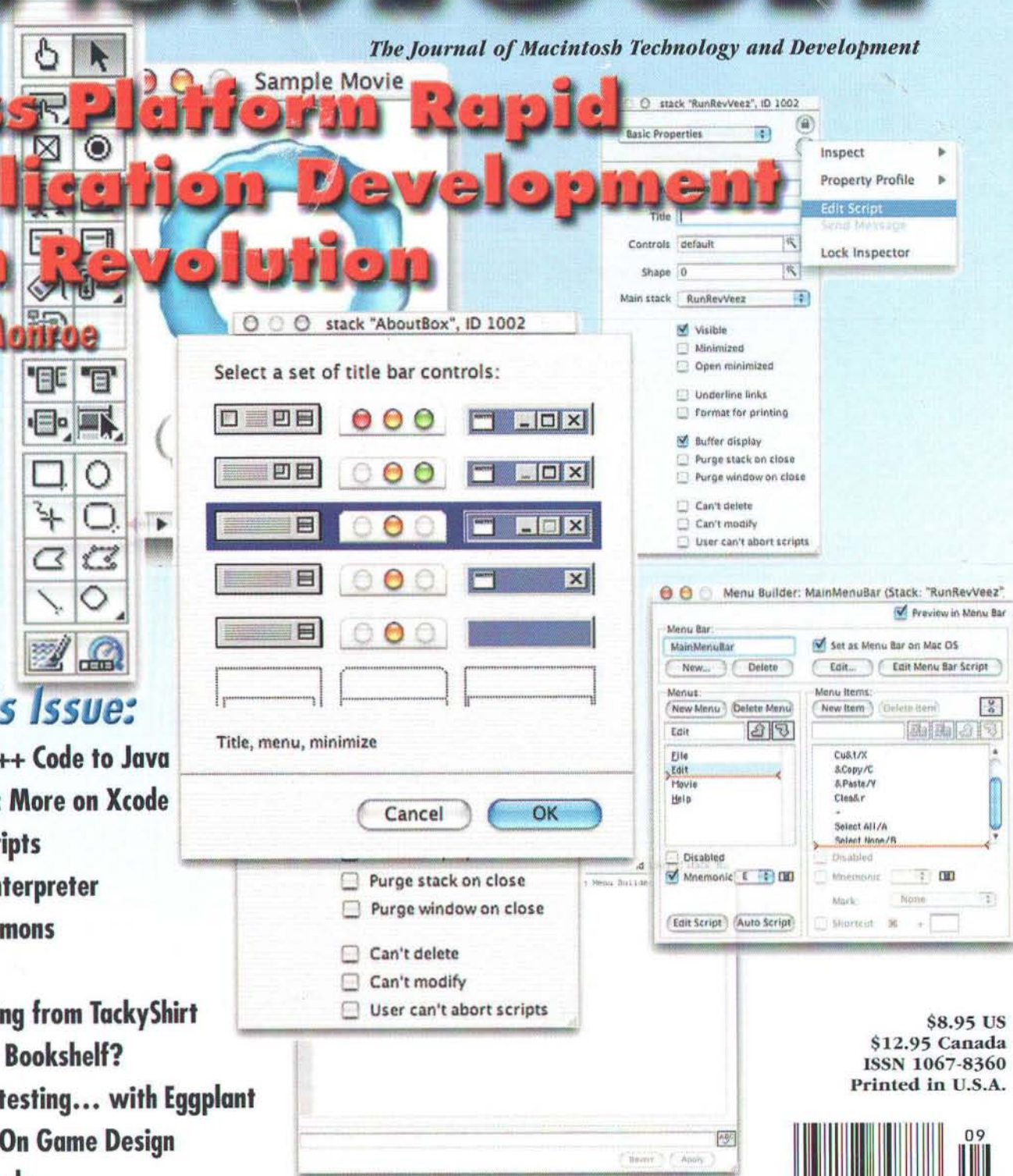
The Journal of Macintosh Technology and Development

Cross Platform Rapid Application Development with Revolution

by Tim Monroe

Also in this Issue:

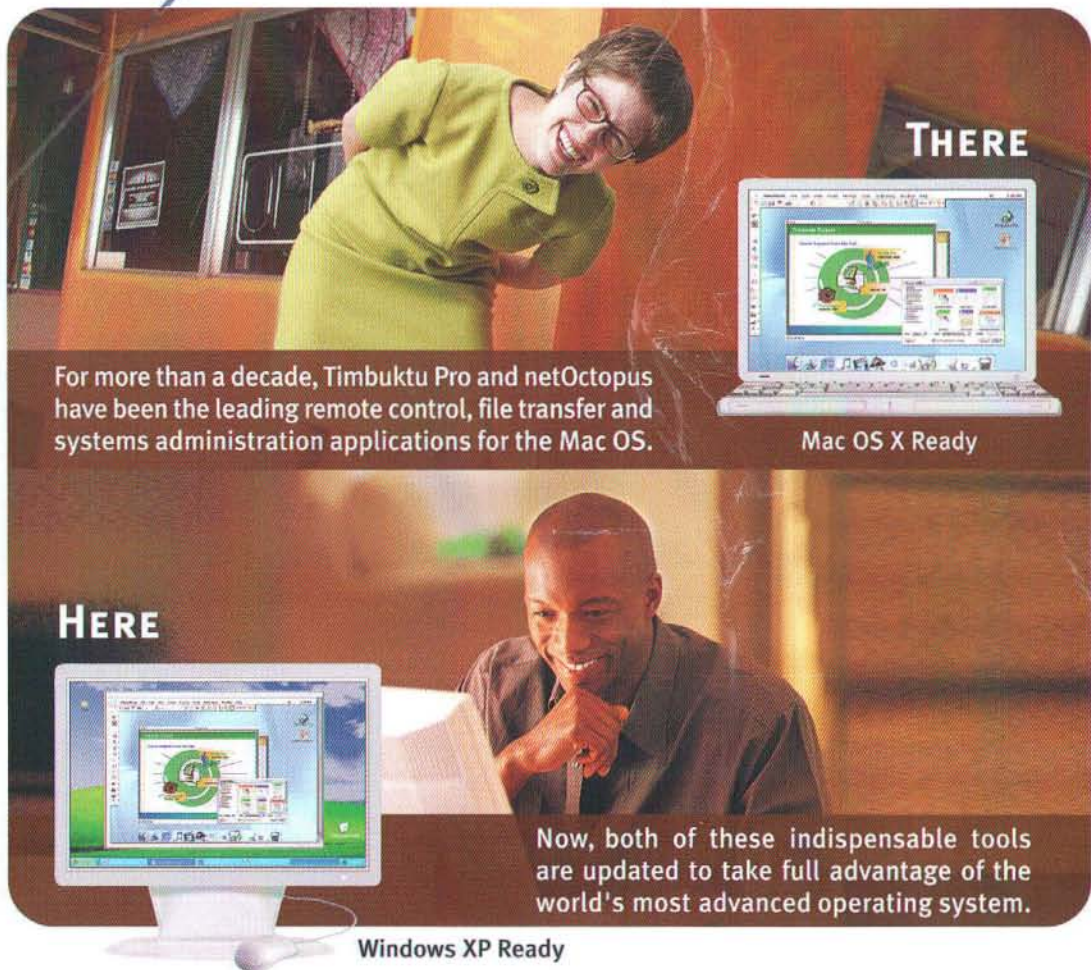
Moving C and C++ Code to Java
Getting Started: More on Xcode
Enabling CGI Scripts
Ch - A C/C++ Interpreter
Time-based Daemons
Juice70
Mac OS X training from TackyShirt
What's on Your Bookshelf?
Automate your testing... with Eggplant
Chris Crawford On Game Design
ARTIS Screen Tools



\$8.95 US
\$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.



Stay In Control Wherever You Go.



THERE

For more than a decade, Timbuktu Pro and netOctopus have been the leading remote control, file transfer and systems administration applications for the Mac OS.

Mac OS X Ready

HERE

Now, both of these indispensable tools are updated to take full advantage of the world's most advanced operating system.

Windows XP Ready

Timbuktu Pro

Whether you're at home or at work, Timbuktu Pro allows you to operate distant computers as if you were sitting in front of them, transfer files or folders quickly and easily, and communicate by instant message, text chat, or voice intercom.

<http://www.timbukutupro.com>

netOctopus

Intuitive and powerful, netOctopus can manage a network of ten or 10,000 computers. Inventory computers, software and devices on your network; distribute software; configure remote computers; and create custom reports on the fly.

<http://www.netoctopus.com>

Learn more, try it, or buy it online. Call us at 1-800-485-5741.



timbuktu® • netOctopus®

netopia.

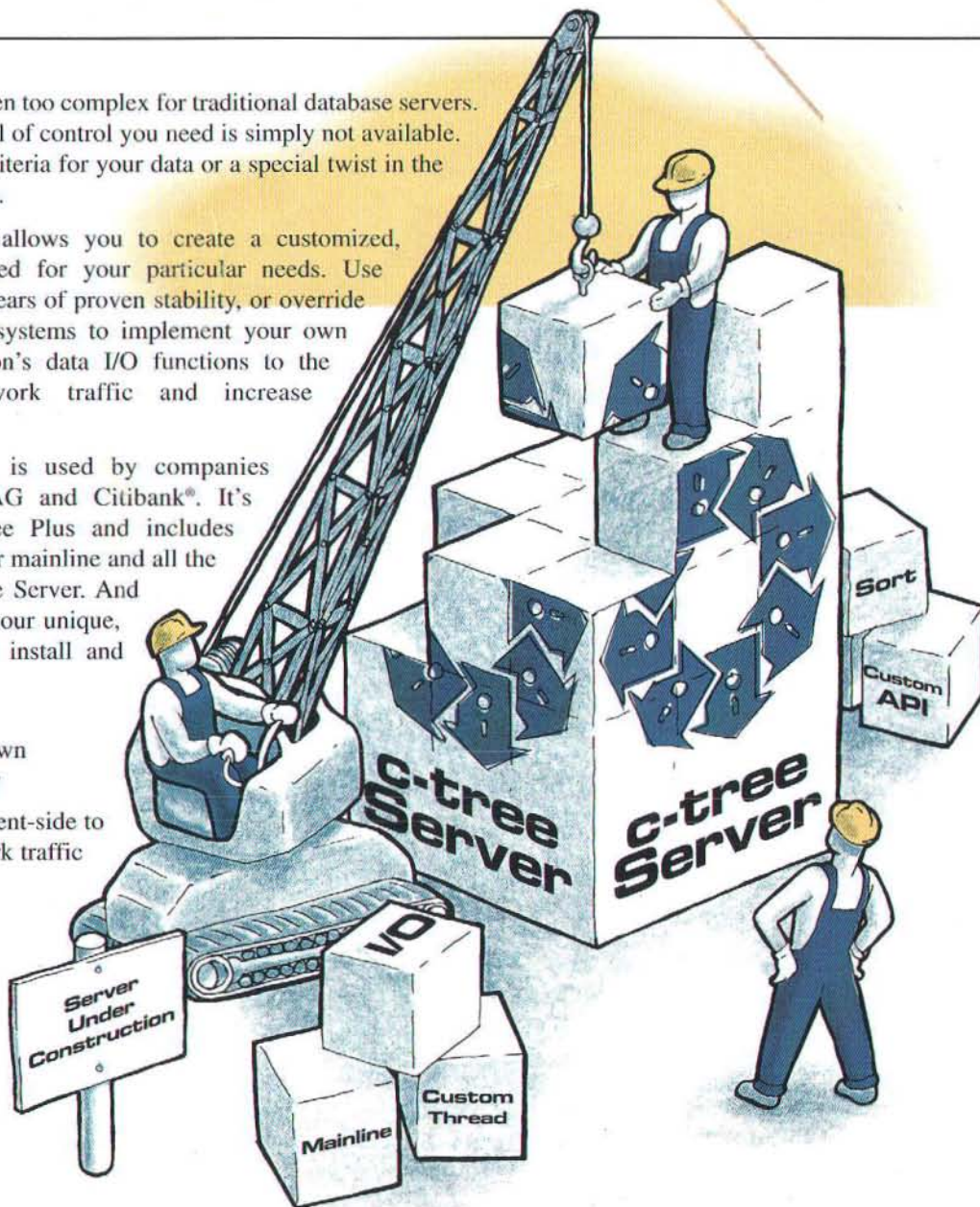
CUSTOMIZE YOUR DATABASE SERVER WITH THE C-TREE® SERVER SDK

Today's database demands are often too complex for traditional database servers. The functionality and precise level of control you need is simply not available. Perhaps you need alternate sort criteria for your data or a special twist in the threading or communication logic.

FairCom's c-tree® Server SDK allows you to create a customized, industrial-strength server designed for your particular needs. Use FairCom's kernel, with over 20 years of proven stability, or override functionality within specific subsystems to implement your own subtleties. Move your application's data I/O functions to the server-side to decrease network traffic and increase performance!

FairCom's c-tree Server SDK is used by companies worldwide such as Software AG and Citibank®. It's integrated seamlessly into c-tree Plus and includes complete source code to the server mainline and all the interface subsystems to the c-tree Server. And best of all, once you've created your unique, customized server, it is easy to install and administer: no DBA required!

- Enhance our server with your own custom server-side functionality
- Move functionality from the client-side to the server-side to reduce network traffic and increase performance
- Modify or replace entire server subsystems
- Complete source for the server mainline, key server subsystems, and client-side
- Flexible OEM licensing



Visit www.faircom.com/ep/mt/sdk today to take control of your server!




FairCom®
www.faircom.com

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

DBMS Since 1979 • 800.234.8180 • info@faircom.com

Other company and product names are registered trademarks or trademarks of their respective owners.

© 2002 FairCom Corporation



Adaptive Server®
Enterprise 12.5
MAC OS X

HELPING FILEMAKER PRO CUSTOMERS SCALE TO NEW HEIGHTS.

Want to enhance the performance of FileMaker Pro on MAC OS X? The same database engine that runs Wall Street can help you do just that.

ASE 12.5 increases the reliability, availability and scalability of your FileMaker Pro application. It provides better data integrity, world-class security and the ability to handle thousands of transactions per minute. It'll also give your users the power of SQL queries.

ASE 12.5 for MAC OS X is yet one more example of how

Sybase is helping today's leading companies achieve Information Liquidity: a highly profitable state where all of your information is transformed into real economic value.

A FREE Developer's Edition download is available online at sybase.com/filemaker.

INFORMATION LIQUIDITY.



BETTER WHEN EVERYTHING WORKS TOGETHER.™

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS

Orders, Circulation, & Customer Service

Press Releases

Ad Sales

Editorial

Programmer's Challenge

Online Support

Accounting

Marketing

General

Web Site (articles, info, URLs and more...)

E-Mail/URL

cust_service@mactech.com

press_releases@mactech.com

ad_sales@mactech.com

editorial@mactech.com

prog_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

http://www.mactech.com

The MacTech Editorial Staff

Publisher • Neil Ticktin

Editor-in-Chief • Dave Mark

Managing Editor • Jessica Stubblefield

Regular Columnists

Getting Started

by Dave Mark

QuickTime Toolkit

by Tim Monroe

Mac OS X Programming Secrets

by Scott Knaster

Reviews/KoolTools

by Michael R. Harvey

Patch Panel

by John C. Welch

Section 7

by Rich Morin

Untangling the Web

by Kevin Hemenway

John and Pals' Puzzle Page

by John Vink

Regular Contributors

Vicki Brown, Erick Tejkowski,
Paul E. Sevinç

MacTech's Board of Advisors

Jordan Dea-Mattson, Jim Straus
and Jon Wiederspan

MacTech's Contributing Editors

- Michael Brian Bentley
- Vicki Brown
- Marshall Clow
- John C. Daub
- Tom Djajadiningrat
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Sun
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Rich Morin
- John O'Fallon, Maxum Development
- Will Porter
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Chuck Von Rospach, Plaidworks

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • Darryl Smart

Marketing Manager • Nick DeMello

Account Executive • Lorin Rivers
adsales@mactech.com • 800-5-MACDEV

Events Manager • Susan M. Worley

International • Rose Kempes

Network Administrator • David Breffitt

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane, Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park,
and Alan Carsrud

All contents are Copyright 1984-2003 by Xplain Corporation. All rights reserved. MacTech and Developer Depot are registered trademarks of Xplain Corporation. RadGad, Useful Gifts and Gadgets, Xplain, DevDepot, Depot, The Depot, Depot Store, Video Depot, Movie Depot, Palm Depot, Game Depot, Flashlight Depot, Explain It, MacDev-I, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, LinuxTech, MacTech Central and the MacTutorMan are trademarks or service marks of Xplain Corporation. Sprocket is a registered trademark of eSprocket Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C o n t e n t s

September 2003 • Volume 19, Issue 9

CODING TRICKS

18 Moving C and C++ Code to Java

By Paul Ammann

GETTING STARTED

6 Getting Started: More on Xcode

By Dave Mark, Editor In Chief

SOFTWARE TESTING

44 Automate your testing... with Eggplant

Exploring a new tool for performing automatic tests

By Dave Kelly

REVIEW: CH

68 Ch - A C/C++ Interpreter

New possibilities for people who like C and Unix.

By Matt Campbell

MAC OS X PROGRAMMING SECRETS

Scott Knaster has been spending time with his mother during her extended illness, and his column will not appear this month. Our thoughts are with Scott and his family.

SECTION 7

12 Time-based Daemons

at(1), batch(1), cron(1), etc.

By Rich Morin

UNTANGLING THE WEB

38 Enabling CGI Scripts

Prepare your server for more powerful dynamic capabilities

By Kevin Hemenway, Omnipotent Yodeler

REVIEWS

16 Juice70

Portable power for most mobile needs

By Michael R. Harvey

32 Mac OS X training from TackyShirt

Training that doesn't suck, really.

By Michael R. Harvey

COVER STORY

50 Revolution

Developing QuickTime Applications with Revolution

By Tim Monroe

BOOK REVIEWS

78 What's on Your Bookshelf?

Ten for X

By Vicki Brown

34 Chris Crawford On Game Design

By Ron Davis

COOL TOOLS

36 ARTIS Screen Tools

Useful utilities for pixel measurements.

By John C. Daub

Copyright 2003 by Dave Mark

Getting Started: More on Xcode

I must say, I am really enjoying digging into Xcode. I'm impressed with what I've seen so far. The bugs I've encountered have mostly been cosmetic and, I assume, easily fixable. My understanding is that the WWDC release is the only Jaguar release and that all fixes and future releases will be Panther-only.

For the moment, I'll stick with the Jaguar version. As soon as a "for public consumption" version of the Panther Xcode makes its way into my hot little hands, I'll shift permanently into Panther mode.

EDITOR WINDOW DETAILS

Last month's column started off by opening an existing Project Builder project called *Sketch.pbxproj* in Xcode and updating the target, creating a new target called *Sketch (Upgrade)*. Go ahead and launch Xcode and open the project. If you are starting from scratch, you'll find *Sketch.pbxproj* in the directory `/Developer/Examples/AppKit/Sketch/`. Remember, you can open a project by navigating to the right directory and clicking the *Open* button. Xcode will find the project file for you.

When the project window appears, click and hold on the *Build and Debug* icon. After a slight pause, a drop-down menu will appear (see **Figure 1**). The little triangle to the lower right of an Xcode tool icon is your clue that there's a drop-down menu lurking within. Just remember to hold and wait a hitch for the menu to appear.



Figure 1. The Build and Debug drop-down menu.

Go ahead and select *Build and Debug* from the drop-down menu. Xcode will build and launch a debug-ready version of *Sketch* and open the debugger window. As you'd expect, the debugging window allows you to control the debugging process.

Use the Dock to bring *Sketch* to the front. Click on the rectangles icon in the palette, then click and drag out a rectangle in the main window (see **Figure 2**). Note that the rectangle that appears is not filled.

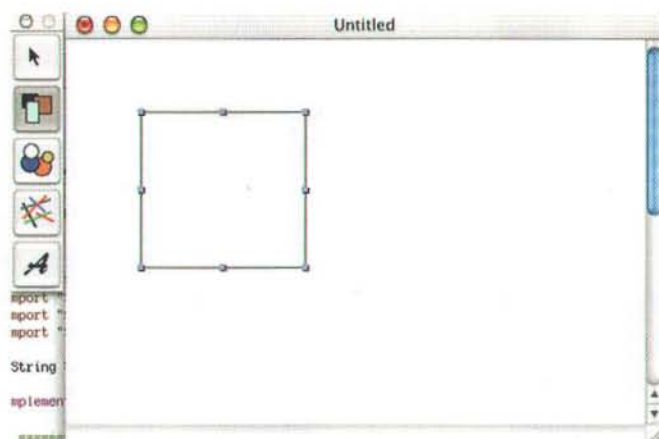


Figure 2. Dragging out a rectangle with Sketch.

Now go back to the project window and click on the Project Symbols group (the last group in the *Groups & Files* column). As you saw last month, this "smart group" lists all the symbols in the project. Type *init* in the search field to reduce the list of symbols (See **Figure 3**).

Dave Mark is a long-time Mac developer and MacTech contributor. Author of more than a dozen books on various Mac-development topics, Dave is all about Xcode these days. Last month's column focused on the basics of opening a Project Builder project, target conversion, and the project window user interface. This month's installment will focus on the editor interface and a demo of fix-and-continue.

VIRUS PROTECTION FOR MACS - EASY IN THE SOPHOS ZONE



Sophos Anti-Virus protects your organization at every level, every point of entry. Our software intelligently recognizes when files need to be virus checked, providing on-access scanning and minimizing the impact on system resources. Every license includes 24x7x365 unlimited technical support as standard and offers comprehensive protection for multiple platforms, including Mac and legacy operating systems. In the Sophos Zone, viruses don't worry Mac users at all.

GET THE FACTS:
Download a **FREE** technical brief,
'Sophos Anti-Virus for Mac OS X'
from www.sophos.com/link/macbrief

www.sophos.com/link/macbrief
Tel 888-216-6703

SOPHOS
anti-virus for business

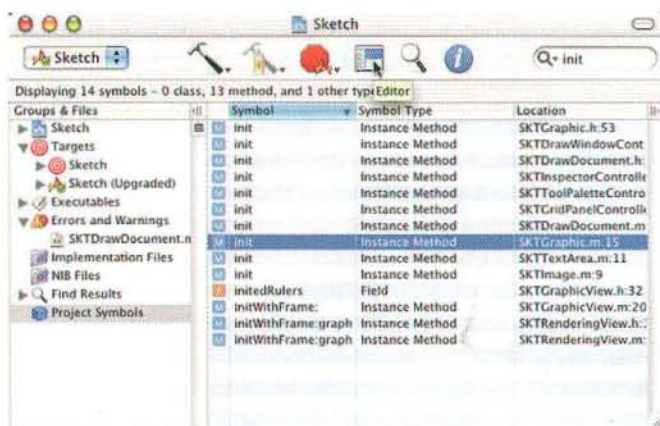


Figure 3. Using the search field to find the *init* method in *SKTGraphic.m*.

We're looking for the *init* method in the file *SKTGraphic.m*. There are two ways to open this file from the listing in the Project Symbols group. We can click on the *Editor* icon in the toolbar (the cursor is pointing to this icon in **Figure 3**) to open an editing pane for that file in the Project Window. Alternatively, we can double-click on the line listing the file we want to open that file in a separate editing window. That's the method we're going to use here.

Figure 4 shows the editor window that appeared when I double-clicked on *SKTGraphic.m*. At the top of the window is the editor toolbar. Just beneath that is the status bar (in **Figure 4** it says *Sketch (Upgraded)*).

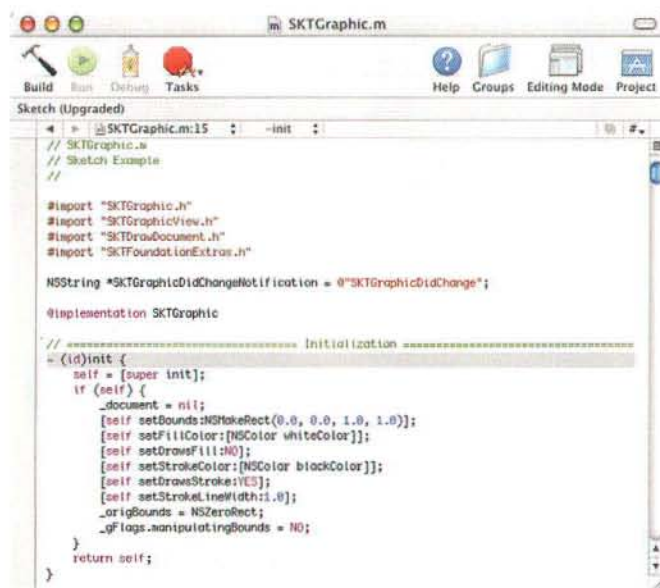


Figure 4. A new editing window listing the file *SKTGraphic.m*.

Below the status bar is the navigation bar. Towards the left edge of the nav bar is a pair of arrows. These act like the left and right arrows in your browser and are used to move between source files. You'll see how this works in a sec.

If you scan to the right side of the nav bar, you'll see an icon showing a pair of overlapping rectangles, a grey one in front of a white one. This is called the counterpart icon. Click on it to switch between a .m and its corresponding .h file. For example, click the counterpart icon to switch to *SKTGraphic.h*. Note that the left arrow on the left side of the nav bar is now enabled. You can click it to go back to the file *SKTGraphic.m*. And, as you'd expect, once you click on the left arrow, the right arrow will be enabled so you can go back to *SKTGraphic.h*. As promised, just like the arrows in your browser.

As you visit different files in your editor window, they are added to a list, sort of a running history. Immediately to the right of the arrows in your nav bar is a popup listing all the files in this history list. Select a file name and the editor edits that file. Makes sense. The tiny dogear icon immediately to the left of the file name is a dirty flag. If the file is dirty (if it has changed since the last save) the icon is grey. Save the file and it returns to white.

Just to the right of the file name is a colon (":") followed by the current line number (either the beginning of a selection, or the line holding the insertion point). If you want to go to a specific line in a file, you can trial and error it, clicking around using the line number in the nav bar to help you home in on the right line. Or you can select *Go to Line...* from the *Find* menu (*Command-L*) and use that interface to select a specific line or range of lines (See **Figure 5**).

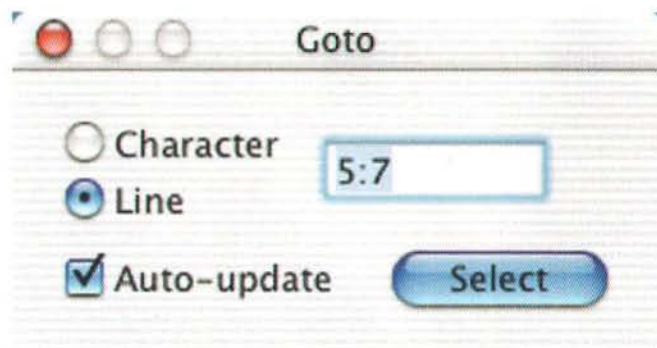


Figure 5. The *Go to Line...* command from the *Find* menu.

Also in the nav bar, just to the right of the file popup menu, lies the function/method popup. As you'd expect, this popup lists all the functions in the current file. Click anywhere inside a function and that function is shown in the popup title. Click on the popup and all the functions/methods in the file are listed in the order in which they appear. Option-click and the functions are listed in alphabetical order, separated by implementation. Give it a try. You'll get it right away.

Next over to the right is the counterpart icon (which we discussed above) followed immediately by the included files popup (it looks like a #) which lets you edit the tree of included files. **Figure 6** shows the included files popup for *SKTGraphic.m*.

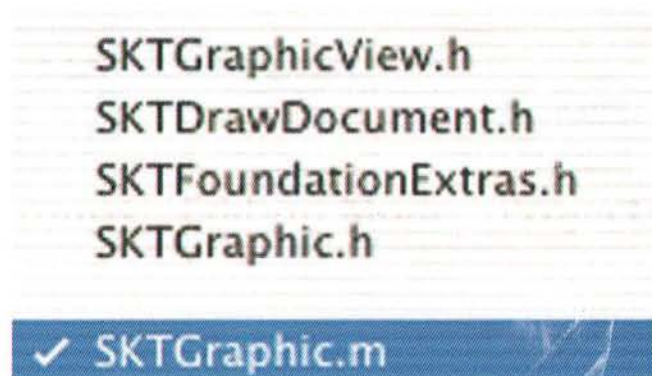


Figure 6. The included files popup menu.

To the lower right of the included files popup (at the top of the scroll bar) is the *split view toggle*. Click on the toggle to create a split in the editing window. Click the toggle again to remove the split. Once you create a split, the *split resize control* appears between the upper and lower scroll bars. To move the split, just drag this control (See **Figure 7**). Note that in the current Jaguar release (and presumably the *only* Jaguar release) of Xcode, the split resize control does not get drawn when you first create a split. Just resize the window a tiny bit and the control will appear.

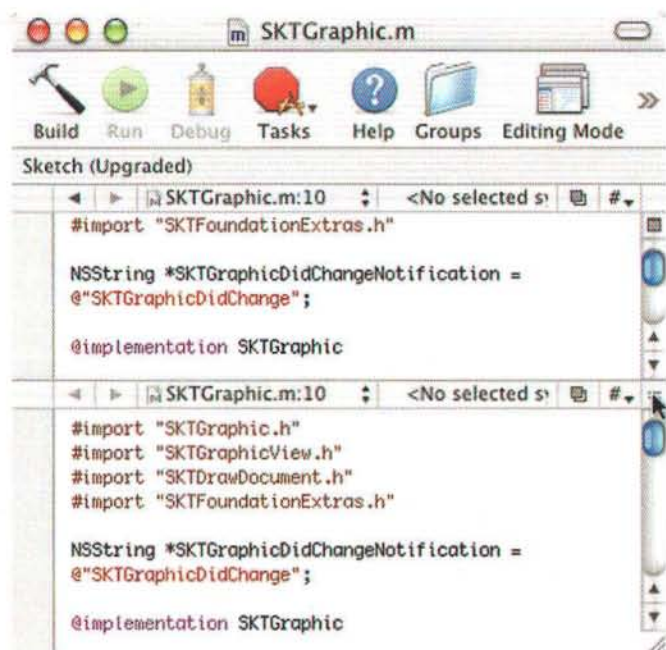


Figure 7. Dragging the split resize control.



Fetch

Mac's best friend.

X
Fetchsoftworks.com

Version 4.0.3 now available.



This and much more available from...

piDog Software

SimpleKeys - piPop - DockSwap - ScreenShot Plus

www.pidog.com

Info@pidog.com

Fix and Continue

One of my favorite Xcode features is Fix-and-Continue. With Mac OS X's built-in support for dynamic linking, you can actually modify your code while it is running and watch the change come to life. Really.

So let's try this out. When you played with Sketch earlier in the column, you dragged out a rectangle and saw that the result was unfilled. If you look through the source code, you'll find that the fill color is white, but the `setDrawFill:` method is called with the parameter `NO`. In *SKTGraphic.m*, here's the `init` method:

```
- (id)init {
    self = [super init];
    if (self) {
        _document = nil;
        [self setBounds:NSMakeRect(0.0, 0.0, 1.0, 1.0)];
        [self setFillColor:[NSColor whiteColor]];
        [self setDrawFill:NO];
        [self setStrokeColor:[NSColor blackColor]];
        [self setDrawsStroke:YES];
        [self setStrokeLineWidth:1.0];
        _origBounds = NSZeroRect;
        _gFlags.manipulatingBounds = NO;
    }
    return self;
}
```

Pay special attention to lines 6 and 7, which correspond to lines 20 and 21 in the file. If you haven't already, launch Sketch in the debugger. You can click on the *Debug* spray can in the editing window or debug window's toolbar, or select from the hammer/spraycan icon in the Project Window's toolbar.

Once Sketch launches in debug mode (you can tell the debugger is active because the *Terminate* and *Pause* icons will be enabled in the *Debug* window), drag out a couple of rectangles, just to prove that they are, indeed, unfilled.

Now go back to your editing window for *SKTGraphic.m* and change line 20 and 21 from:

```
[self setFillColor:[NSColor whiteColor]];
[self setDrawFill:NO];
```

to:

```
[self setFillColor:[NSColor greenColor]];
[self setDrawFill:YES];
```

Do *NOT* save. You'll see why in a sec.

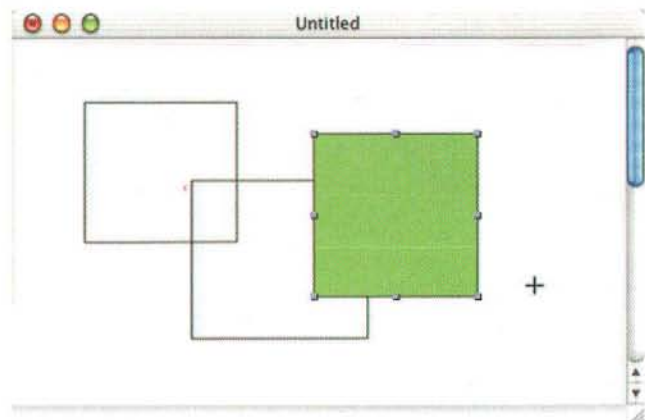
Basically, we've changed the fill color from white to green and we've told Sketch that we *do* want our shapes filled.

Now comes the cool part.

Select *Fix* from the *Debug* menu.

Remember not to save!

You'll see a message in the Debug window about compiling and building. Once that settles down, go back to the running Sketch and click and drag out a rectangle. Wait, I've done it for you. Check out **Figure 8**. As you were expecting, the rectangles are now filled and green.



Once you are done playing with your new green rects, quit Sketch and rerun it, either as a straight run or via the debugger. Notice that your rectangles are back to being unfilled (you did remember to *not* save your changes, right?) To me, this is a beautiful way to play with your code. You can make changes, experiment, use *Debug/Fix* to test out your changes, see what works for you, then save what you want to.

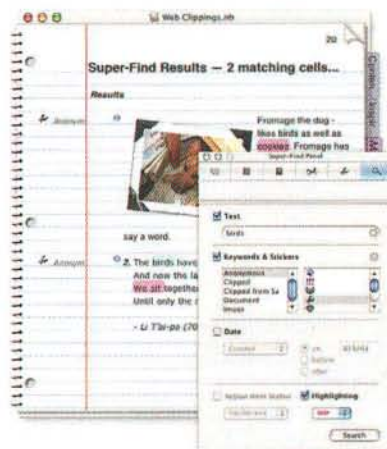
You may have noticed a scotch-tape dispenser icon in the *Debug* window with the label *Fix*. Clicking this icon does the same thing as selecting *Fix* from the *Debug* menu. There is a bug (I believe it is fixed in the current Panther release) where the icon is disabled (it just beeps when you click it). Fortunately, the bug does *not* affect the *Fix* item in the *Debug* menu.

MacTech®

Get MacTech delivered to your door at a price **FAR BELOW** the newsstand price. And, it's **RISK FREE!**

Subscribe Today!
www.mactech.com

Never ask “Now where did I put that...” again.



Introducing Circus Ponies™ NoteBook



The \$50 valet for your digital life

Wouldn't it be great to have a personal assistant to keep your stuff **organized**?

Now you can.

Photos, e-mails, sound files, images, documents. NoteBook keeps track. While you're busy taking notes and clipping information, NoteBook **indexes** it all. Whatever you can remember, NoteBook can find.

machOME Our customers love
NoteBook. We think
you will too. Visit circusponies.com for
a demo. And hire that assistant after all.

circusponies.com
415.695.3100



© Copyright 2003 Circus Ponies Software, Inc. All Rights Reserved. Circus Ponies, the Circus Ponies logo, NoteBook, and the NoteBook logos and icons are trademarks of Circus Ponies Software, Inc.

By Rich Morin

Time-based Daemons

at(1), batch(1), cron(1), etc.

Most daemons (i.e., background processes) are event-based, responding to an incoming packet, the appearance of a file, etc. Some daemons are time-based, however, occurring at a given time (or times).

BSD (and thereby OSX) makes it very easy to set up time-based daemons. In fact, there are several ways to do this, depending on your needs. Let's look at some of the options.

CRON

The `cron(8)` subsystem (see also `crontab(1,5)`) runs commands at times which are specified in one or more control files. It is also, as described below, the basis for time-based services such as `at(1)` and `periodic(8)`.

Note: The command "`man cron`" fails in Mac OS X 10.2.6, but you can view the man page by typing "`more /usr/share/man/cat8/cron.8.gz`".

Originally, there was only one `crontab(8)` file, located at `/etc/crontab`. This file was only editable by root, though it could run commands as other users. Later, individual users were given the ability to maintain their own control files, using the `crontab(1)` command.

The default version of `/etc/crontab` on Mac OS X looks like:

```
# /etc/crontab
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
HOME=/var/log
#
#minute hour mday month wday who  command
#
#*/5 * * * * root /usr/libexec/atrun
#
# Run daily/weekly/monthly jobs.
15 3 * * * root periodic daily
30 4 * * 6 root periodic weekly
30 5 1 * * root periodic monthly
```

Environment variables can be set (using Bourne shell syntax) for the scheduled commands. Thus, the commands listed in this file will have `HOME`, `PATH`, and `SHELL` set for them. Be

sure to take these settings into account when writing scripts to be run under cron; if your script asks for a command that isn't found on the `PATH`, for example, it won't act as desired.

As in the case of most BSD control files (and many scripting languages), "`#`" can be used to indicate the start of a comment, extending through the end of the current line. This is often used to disable scheduling lines (such as the one for "`atrun`", in this example).

Each scheduling line has three parts, separated by white space. The first part may be a special string (e.g., `@reboot`, `@daily`), but more commonly it will be a set of five fields, also separated by white space. The second part is the username (e.g., `root`) under which the command will be run. The third part is the command (e.g., "`periodic daily`").

In the example, "`periodic daily`" is scheduled to be run (as `root`) at 3:15 AM every day. Similarly, "`periodic weekly`" and "`periodic monthly`" are scheduled for 4:30 AM each Saturday and 5:30 AM on the first day of each month, respectively.

The format of `crontab` files for individual users is almost identical to that for `/etc/crontab`. The only difference is that the "who" (username) field is not present. This makes sense; only the `root` account is able to set the user id under which a command will be run.

Although `/etc/crontab` can be edited in any desired manner, the individual `crontab` files must be edited by means of the `crontab(1)` command. This prevents race conditions, ensures that the `cron` daemon will notice any changes, etc.

PERIODIC

If you have a system maintenance command that needs to be run during off hours on a daily, weekly, or monthly basis, the periodic subsystem (`periodic(8)`, `periodic.conf(5)`) may be exactly what you want.

The `periodic(8)` command is actually a shell script. I won't discuss it here, but you may wish to give it a look: "`more /usr/sbin/periodic`". Basically, however, the script runs every executable file found in the specified directory. For example, "`periodic daily`" runs any commands found in `/etc/periodic/daily`:

```
% wc -l /etc/periodic/daily/*
56 /etc/periodic/daily/100.clean-logs
```

Rich Morin has been using computers since 1970, Unix since 1983, and Mac-based Unix since 1986 (when he helped Apple create A/UX 1.0). When he isn't writing this column, Rich runs Prime Time Freeware (www.ptf.com), a publisher of books and CD-ROMs for the Free and Open Source software community. Feel free to write to Rich at rdm@ptf.com.

Multiple formats. Multiple platforms. Complex installers.

Aladdin solves the compression and installation puzzle.

**Trying to figure out how to handle multiple
compression formats and platforms?**

The StuffIt Engine solves the compression puzzle.



Aladdin's StuffIt Engine SDK:

- Adds value to your application by integrating powerful compression and encryption.
- Is the only tool that supports the StuffIt file format.
- Provides a single API that supports over 20 compression and encoding formats common on Macintosh, Windows, and Unix.
- Makes self-extracting archives for either Macintosh or Windows.
- Available for Macintosh, Windows, Linux, or Solaris.

Licenses start as low
as \$99/year

To learn more, visit:
www.stuffit.com/sdk/

StuffIt Engine SDK™ The power of Stuffit in your software.



**Looking for the easiest and fastest
way to build an installer?**

StuffIt InstallerMaker completes your puzzle.

It's not enough just to write solid code anymore. You still have to write an installer for your users. StuffIt InstallerMaker makes it simple and effective.

- StuffIt InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.
- Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers.
- Make demoware in minutes. Create Macintosh OS X and Macintosh Classic compatible installers with StuffIt InstallerMaker.

Prices start at \$250

To learn more, visit:
[www.stuffit.com/
installermaker/](http://www.stuffit.com/installermaker/)

StuffIt InstallerMaker™ The complete installation solution.™



www.stuffit.com
(831) 761-6200

© 2002 Aladdin Systems, Inc. Stuffit, StuffIt InstallerMaker, and StuffIt Engine SDK are trademarks of Aladdin Systems, Inc. The Aladdin logo is a registered trademark. All other products are trademarks or registered trademarks of their respective holders. All Rights Reserved.

```
131 /etc/periodic/daily/500.daily
187 total
```

I wouldn't suggest modifying any of these files, as Apple may overwrite them in an update, but putting in your own files should be fairly safe. Just stuff an executable file into the appropriate subdirectory, picking the filename to sort into the desired execution order. For example, if you have a script that needs to run after "500.daily", you could name it something like "600.local.fooscript".

You may enjoy looking through these files to see what gets done while you're off snoozing: try "more /etc/periodic/*/*". The configuration files, described in `periodic.conf(5)`, are also worth a look.

AT, BATCH, ETC.

The `at(1)` and `batch(1)` commands act in a very similar manner to each other. Both commands schedule a file for execution at a specified time. The difference is that `batch(1)` also checks the system load level, ensuring that the command doesn't add work to an already-overloaded system.

In order to use either command, however, you'll have to uncomment the "atrun" line in `/etc/crontab`, causing the program to be run every five minutes:

```
* /5 * * * * root /usr/libexec/atrun
```

Actually, there's no particular reason why you couldn't schedule `atrun` to run every minute, if you wish. On a desktop machine, an occasional process start-up is unlikely to make a noticeable difference. To try this, just edit the line to:

```
* * * * * root /usr/libexec/atrun
```

Note: The documentation and configuration of `at(1)` in OS X 10.2.6 are a bit deficient. Although the `at(1)` man page says that "Traditional access control to `at` and `batch` via the files `/var/at/at.allow` and `/var/at/at.deny` is not implemented", the program will fail unless (at least) one of these files is present. The spool directory (`/var/at/spool`) may also be missing, causing scheduled jobs to silently fail. Fortunately, the fixes are simple:

```
% su
Password:
# touch /var/at/at.deny
# mkdir /var/at/spool
# exit
exit
%
```

Having worked our way past the setup hassles, let's try running some `at(1)` jobs. Here's a short test script we can use:

```
:
# at - at(1) test script

(date; printenv | sort) > att.$$out
```

For the shell-challenged, here's a rundown of what's going on here. The initial colon tells the kernel that the script should be interpreted by the Bourne shell. The real work is done by a single line which starts up a subshell (subsidiary copy of the shell), has it run "date" and "printenv | sort", and redirects the (concatenated) output into a file.

Because "\$\$" evaluates to the process ID of the interpreting shell, the name of the file will look something like "att.12345.out". After you have edited the file, make it executable, run it, and examine the results:

```
% chmod +x att
% att
% more att.$$out
Fri Jul 4 18:46:38 PDT 2003
HOME=/Users/rdm
PATH=/Users/rdm/bin:...
PWD=/Users/rdm/...
SHELL=/bin/tcsh
...
```

The output shows us the date and time that the command was run, as well as the settings for any environment variables. Now, let's try scheduling the script via `at(1)`, waiting for it to get run, and comparing the output with that of our first (manual) run:

```
% at -f att +1 minute
Job a010ce73c.000 will be executed using /bin/sh
% atq
Date           Owner      Queue    Job#
19:04:00 07/04/03   rdm      a      a010ce73c.000
...
% atq
% diff att.$$out
_ i _ c _ l _
_ < _ F _ r _ i _ J _ u _ l _ _ 4 _ 1 _ 9 _ : _ 0 _ 0 _ : _ 4 _ 8 _ _ P _ D _ T _ _ 2 _ 0 _ 0 _ 3 _
_ - _ - _ - _
_ > _ F _ r _ i _ J _ u _ l _ _ 4 _ 1 _ 9 _ : _ 0 _ 5 _ : _ 0 _ 0 _ _ P _ D _ T _ _ 2 _ 0 _ 0 _ 3 _
_ 2 _ 3 _ _ 2 _ 5 _ c _ 2 _ 3 _
_ < _ S _ H _ L _ V _ L _ = _ 2 _
_ < _ T _ E _ R _ M _ = _ v _ t _ 1 _ 0 _ 0 _
_ < _ T _ E _ R _ M _ C _ A _ P _ = _ - _ - _ - _
_ - _ - _ - _
_ > _ S _ H _ L _ V _ L _ = _ 1 _
```

Not too many changes, really. The time changed, of course, but most of the environment variables stayed the same. `SHLVL` (the shell level) is lower for the `at(1)` run, because no interactive shell was involved. The `TERM` and `TERMCAP` variables aren't set for the `at(1)` run, because no terminal is attached to the process.

ROLLING YOUR OWN

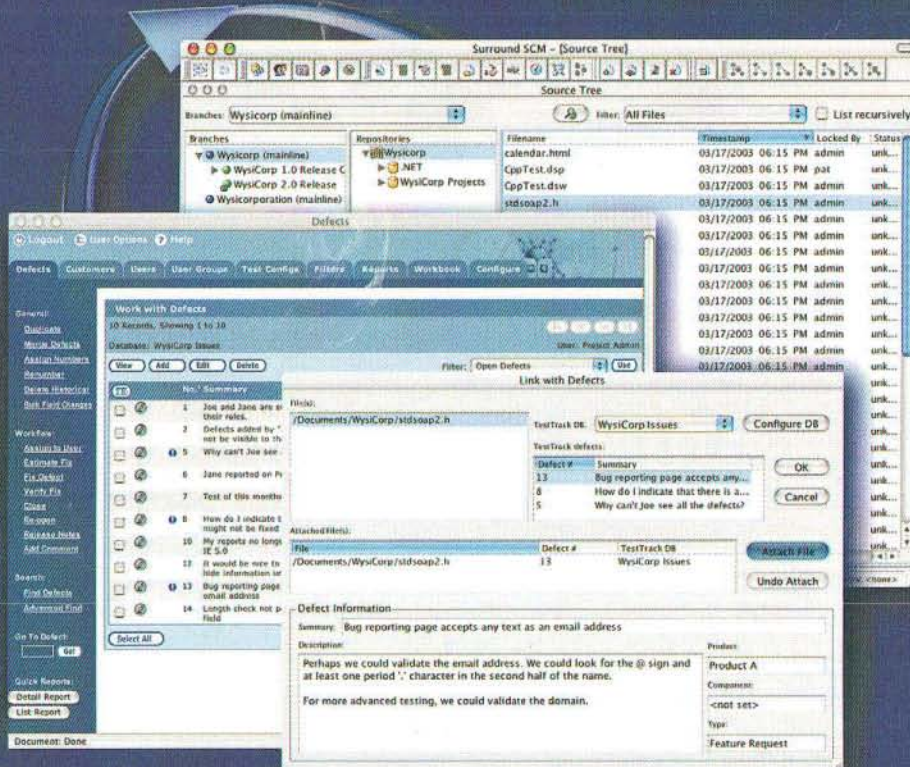
If none of these facilities is quite what you need, consider creating your own time-based daemon. Simply putting a process to sleep for a specified period is quite simple; making a process wake up at a specified time is a bit trickier, but still quite possible.

If you take this approach, however, you may want to look at the source code for existing routines that perform similar services. The Darwin source code (www.opendarwin.org) has the source code for the commands described in this column. The CPAN (cpan.perl.org) is a good place to look for relevant Perl modules.

Complete Source Control and Defect Management

Seapine Software™
changing the world
of software development

for Mac OS X



Effective source code control and defect tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- Fast and secure remote access to your source files and defects — work from anywhere
- Advanced branching simplifies managing multiple versions of your products
- Link code changes with defects and change requests — know who changed what, when, and why
- Scalable and reliable cross-platform, client/server solutions support Mac OS X, Windows, Linux, and Solaris
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget

Seapine Software Product Lifecycle Management
Award winning, easy-to-use software development tools

Seapine
Surround SCM
Seapine
TestTrack PRO



**Download Surround SCM
and TestTrack Pro at
www.seapine.com
or call 1-888-683-6456**

all product names listed herein are registered trademarks of their respective owners. All rights reserved.



By Michael R. Harvey, Reviews Columnist

Juice70

Portable power for most mobile needs

Today we talk about juice. There's all kinds. Orange juice, grape juice, cranberry juice, and my sons favorite, the ubiquitous juice box. But that's not important right now. The Juice we have before us today is a highly adaptable and expandable power supply. The Juice70, from iGo Mobility Products, is an adapter that can be used with most laptops, Apple, and PC. The basic package comes with the main power brick, an interchangeable power cord for 110 outlets, as well as one that plugs into both auto power outlets and power outlets found in airline seats. On the other end of the power brick is the cord that plugs into your computer. For this side, several tips are provided that fit to most laptop computers, as well as a chart to help you determine which goes into which. Be careful not to lose the card, as some adapters can fit into the wrong laptop, potentially causing damage. A carrying case (of fair quality) is included to hold all this stuff. That is not all, though.

Part of what makes the Juice70 different from other third party adapters is what iGo terms the "Peripheral Powering System." On the cord from the power brick to the computer is a small port. Into this port, you can plug any of a number of adapters (all of which are sold separately) for cell phones, Nokia, Kyocera, and others, and PDAs from the likes of Sony, Compaq, and others. The compatibility chart for these can be found on the iGo web site.

This is really one great product. The only knock is against it is the cords. They can get unwieldy at times. If they had some better way to be controlled, better than the included Velcro straps anyway, or were retractable, that would make this product near perfect. The Juice70 is \$119.95 directly from iGo. Add on charging cables for PDAs and cell phones cost \$19.99 each. This is good way to give yourself that extra charger you always seem to need.

www.igo.com



MacTech[®]
M A G A Z I N E

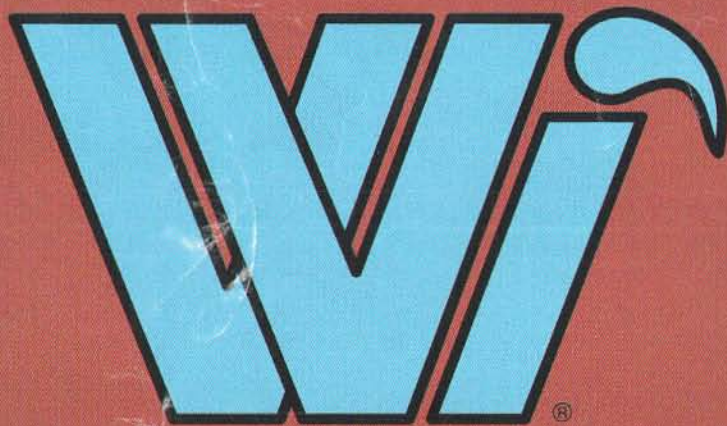
Get MacTech delivered to your door at a price
FAR BELOW the newsstand price.

And, it's **RISK FREE!**

Subscribe Today!
www.mactech.com

TECHNICAL DIAGRAMS AND REPORTS

PLOTS • GRAPHS • CHARTS • VISUALIZATION
TEXT • TABLES • FORMS



CONTRACTING

SPECIALIZING IN BUSINESS AND TECHNICAL REPORTING
DATA FROM ANY SOURCE

WHY SETTLE FOR LESS WHEN YOU CAN HAVE THE TOP-MOST EXPERTS?

PROVIDING BUSINESS, ENGINEERING AND SCIENTIFIC REPORT SYSTEMS SINCE 1989

Vvidget Pro™:

A programming kit to control and report data in visual formats.

Peer Visual™:

A server to distribute dynamic visuals to the web and printers.

Vvidget Fusion™:

A distributed and parallel network data acquisition kit.

Vvidget Signals™:

A real-time multi-channel strip chart and signal display kit.

Vvidget User™:

A graph building tool, palette, and preconfigured graphing server.

Limited Time Offer: \$89 (\$49 for students) (this ad made with Vvidget User)

- Technical Analysis
- Real-Time Trading
- Laboratory Instruments
- Process Control
- Database Interfaces
- Modeling
- Device Drivers
- Data Archival
- Mac® OS X Experts

VVI®

www.vvi.com

info@vvi.com

888-VVI-PLOT

by Paul Ammann

Moving C and C++ Code to Java

Even though Java is catching on pretty rapidly as a powerful new language with a lot of potential, there is a significant problem in dealing with the large existing C and C++ code base. As a Java programmer, you have two solutions to this problem: convert your existing C/C++ code to Java or interface your Java code to the C/C++ code. Although the latter solution is certainly a viable option, especially in terms of development resources, this article focuses on the former. If you're really bent on keeping your C/C++ code as it is, I will cover this topic in my next article, "Interfacing to Existing C and C++ Libraries."

The prospect of moving C/C++ code to Java might seem somewhat daunting at first. However, the similarities in syntax between the languages alone makes matters a little easier to handle. In reality, you can isolate the differences between the languages and form a strategy aimed at fixing trouble spots in the code.

This article takes on the challenge of converting C/C++ code to Java code an issue at a time. You learn not only how to convert C/C++ code to Java, but also why Java is different and requires the changes in the first place. If you are following along attempting to port your own code, try to make incremental changes as you cover each section. You'll probably find that the task isn't as bad as you originally thought. With that in mind, let's get started!

FILE ORGANIZATION

Before you even begin changing any source code, it's important to understand a fundamental difference between source files in Java and C/C++. In C and C++, most source files come in pairs consisting of a header file (.h) and an implementation file (.c or .cpp). The purpose of this file structure is to separate the declarations of the functions and classes from the definitions. This enables other programmers to understand code by viewing header files, while keeping them out of the specific implementation details found in the implementation files.

In Java, there is only one source file (.java) per logical code structure. Java classes contain class declaration

information that can be easily extracted using the javap class file disassembler tool that comes with the JDK. Because of this, there is no need to maintain a header file with class declaration information. All the code for a Java class goes in the .java source file.

What does this mean to you? Well, it means you should get ready to merge all your header and implementation files into .java files. Once you've done that, you can move into modifying the code itself.

THE PREPROCESSOR

All C/C++ compilers implement a stage of compilation known as the preprocessor. The C++ preprocessor basically performs an intelligent search and replace on identifiers that have been declared using the `#define` directive or `typedef`. Although most advocates of C++ discourage the use of the preprocessor, which was inherited from C, it is still widely used by most C++ programmers. Most of the processor definitions in C++ are stored in header files, which complement the actual source code (implementation) files.

The problem with the preprocessor approach is that it provides an easy way for programmers inadvertently to add unnecessary complexity to a program. Many programmers using `#define` and `typedef` end up inventing their own sublanguage within the confines of a particular project. This results in other programmers having to go through the header files and sort out all the `#define` and `typedef` information to understand a program, which makes code maintenance and reuse almost impossible. An additional problem with the preprocessor approach is that it is very weak when it comes to type checking and validation.

Java does not have a preprocessor. It provides similar functionality (`#define`, `typedef`, and so forth) to that provided by the C++ preprocessor, but with far more control. Constant data members are used in place of the `#define` directive, and class definitions are used in lieu of `typedef`. The end result is that Java source code is much more consistent and easier to read than C++ source code. Additionally, as you learned earlier, Java programs don't use header files; the Java compiler builds class declarations

Paul Ammann works as a Network Security Engineer for a large logistics company. He speaks UNIX as a second language, and likes writing programs as a hobby. When he isn't sitting in front of a computer, he is spending time with his wife Eve.

directly from the source code files, which contain both class declarations and method implementations.

Let's look at an example; **Listing 1** shows a C++ header file for a ball class.

Listing 1: The C++ ball class

```
#define COLOR_RED 1
#define COLOR_YELLOW 2
#define COLOR_BLUE 3
#define MATERIAL_RUBBER 1
#define MATERIAL_LEATHER 2

class ball {
    float diameter;
    int color;
    int material;
};
```

To move this code to Java, the only change is to get rid of the preprocessor `#define` directives and the semicolon at the end of the class declaration. You get rid of the `#define` directives by declaring Java class members that are `static` and `final`. For data members in Java, the `static` keyword means there is only one copy for the entire class, and the `final` keyword means that they are constant, which is usually the motive for using `#define` in C/C++ code. **Listing 2** shows the resulting Java version of this class. Keep in mind that the Java version is not stored in a header file, because Java doesn't support header files; in Java, definitions and declarations are combined in one place, the `.java` source file.

Listing 28.2: The Java ball class.

```
class ball {
    // Constants
    static final int COLOR_RED = 1;
    static final int COLOR_YELLOW = 2;
    static final int COLOR_BLUE = 3;
    static final int MATERIAL_RUBBER = 1;
    static final int MATERIAL_LEATHER = 2;

    // Variables
    float diameter;
    int color;
    int material;
}
```

The Java version of `ball` pulls all the constants inside the class definition as `static final` integers. Within this class, you would then refer to them just as you would the previous C++ versions. However, outside this class they are inaccessible, because they have been left at their default access type. To make them visible by other classes, you simply declare their access type as `public`, as shown in **Listing 3**. The statement about default access isn't entirely true; you learn the whole scoop about access types later in this article.

Listing 3: The Java ball class with public constants.

```
class ball {
    // Constants
    public static final int COLOR_RED = 1;
    public static final int COLOR_YELLOW = 2;
```

```
public static final int COLOR_BLUE = 3;
public static final int MATERIAL_RUBBER = 1;
public static final int MATERIAL_LEATHER = 2;
```

```
// Variables
float diameter;
int color;
int material;
}
```

In this version of `ball`, the constants are readily available for other classes to use. However, those classes must explicitly refer to the constants using the `ball` class name:

```
int color = ball.COLOR_YELLOW;
```

STRUCTURES AND UNIONS

There are three types of complex data types in C/C++: classes, structures (structs), and unions. Java supports only one of these data types, classes. Java forces programmers to use classes when the functionality of structures and unions is desired. Although this sounds like more work for the programmer, it actually ends up being more consistent, because classes can imitate structures and unions with ease. Furthermore, supporting structs and unions would have put a

audio recording • editing • effects

Felt Tip
Sound Studio
2.1

FELT TIP SOFTWARE presents a MAC OS X application "SOUND STUDIO" available on CD-ROM and as a WEB DOWNLOAD sales by KAGI written by LUCIUS KWOK © 2002 Felt Tip Software. All rights reserved.

www.felttip.com/ss

major hole in the whole concept of the Java language being object-oriented. The Java designers really wanted to keep the language simple, so it only made sense to eliminate aspects of the language that overlapped.

Converting structs and unions to Java classes is pretty easy. Take a look at **Listing 4**, which contains a polar coordinate C struct.

Listing 4: The C polar struct.

```
typedef struct polar {  
    float angle;  
    float magnitude  
} POLAR;
```

Notice that this struct uses a `typedef` to establish the polar type. As you learned earlier, `typedefs` aren't necessary in Java, because everything is an object with a unique type. Java doesn't support the concept of a struct either. **Listing 5** contains the Java version of the polar class.

Listing 5: The Java polar class.

```
class polar {  
    float angle;  
    float magnitude  
}
```

In addition to changing the `typedef struct` declaration to class, notice that the Java polar class isn't followed by a semicolon. This is a small, but often overlooked, difference

between Java and C++; semicolons aren't necessary in Java class definitions.

FUNCTIONS AND METHODS

In C, code is organized into functions, which are global subroutines accessible to a program. C++ added classes and in doing so provided class methods, which are functions that are connected to classes. C++ class methods are very similar to Java class methods. However, because C++ still supports C, there is nothing keeping C++ programmers from using functions. This results in a mixture of function and method use that makes for confusing programs.

Java has no functions. Being a purer object-oriented language than C++, Java forces programmers to bundle all subroutines into class methods. There is no limitation imposed by forcing programmers to use methods instead of functions. As a matter of fact, implementing subroutines as methods encourages programmers to organize code better. Keep in mind that, strictly speaking, there is nothing wrong with the procedural approach of using functions; it just doesn't mix well with the object-oriented paradigm that defines the core of Java.

Because almost all C/C++ code contains some degree of function use, this is a particularly important issue when porting C/C++ code to Java. Fortunately, it's mostly an organizational change. The whole point of functions is to move code into a logically separate procedure that can be called from the main program or other functions. You can easily recreate this scenario in Java without having to "objectify" the code completely. The solution is to move global C/C++ functions into method-only organizational Java classes. Check out **Listing 6**, which contains a series of string encryption/decryption function prototypes.

Listing 6. The C string encryption/decryption function prototypes.

```
char EncryptChar(char c, int key);  
char DecryptChar(char c, int key);  
char* EncryptString(const char* s, int key);  
char* DecryptString(const char* s, int key);
```

These functions are global C functions that encrypt and decrypt characters and strings. Of course, in C/C++ there is no pure concept of a string; an array of characters is the best you get (more on that later in this article). A straight function-to-method port of these functions in Java is shown in **Listing 7**.

Listing 7: The Java string encryption/decryption methods encapsulated within the crypt class.

```
class crypt {  
    public static char encryptChar(char c, int key) {  
        // character encryption code  
    }  
  
    public static char decryptChar(char c, int key) {  
        // character decryption code  
    }  
}
```

stockicons.com

brought to you by the Iconfactory

Icons to go!

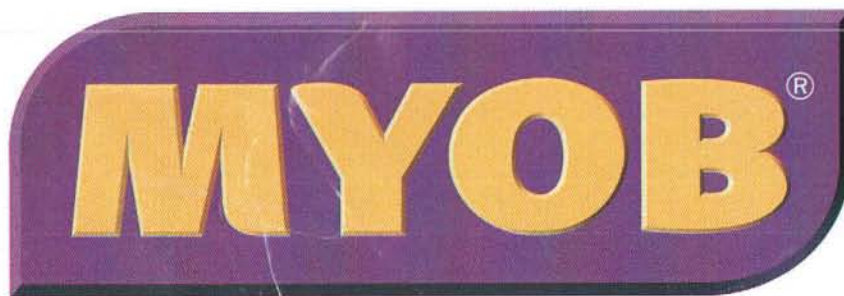


Now developers can purchase complete collections of professionally designed icons at an affordable price from the premier source for custom designed icons in the industry. See some of the work we have done for Apple, Microsoft, Aladdin, Intuit, Palm and many others at www.iconfactorydesign.com.

\$349.00

Each stock icon collection contains 80 individual icons in three pixel sizes - 32x32, 24x24 and 16x16. Collections are unique in style and are provided in an array of formats including transparent TIFFs, GIF, PNG, .icns, and Windows format .ico's.

For more information visit www.stockicons.com



MYOB AccountEdge
Evolving with the Mac since 1989
Small Business Management and Accounting

www.myob.com/us



800-322-MYOB (6962)

Small Business. Smart Solutions.™

© MYOB 2003

```

}

public static String encryptString(String s, int key) {
    // string encryption code
}

public static String decryptString(String s, int key) {
    // string decryption code
}
}

```

In Java, you have to package the functions as methods in a class, `crypt`. By declaring them as `public static`, you make them readily available to the entire Java system. The key aspect of the Java version of the functions is that their implementations are defined in a Java class because Java doesn't support the header/source file organization. All class information goes directly into the class definition. Notice that the standard naming convention for Java methods is to begin each method name with a lowercase character. To use the Java methods, you have to reference them with the `crypt` class name:

```
char c = crypt.encryptChar('a', 7);
```

The only other change to the C functions is the usage of `String` objects rather than `char` pointers because Java doesn't support pointers. You get into more details surrounding strings and pointers a little later in this article.

PROCEDURAL-TO-OOP CONVERSION

Although the Java `crypt` class provides working Java versions of the procedural C functions, performing this type of conversion isn't always enough. The `crypt` class provides a good example of how you can maintain a procedural feel within a Java class. Java is an object-oriented language, however, and you should design your code to fit into the object-oriented paradigm whenever possible. The `crypt` class is no exception to this rule.

Examining the `crypt` class methods, it is apparent that some things could be modified to make the class fit into a more object-oriented design. **Listing 8** contains the source code for the revised, objectified `crypt` class.

Listing 8. The revised Java >crypt class.

```

class crypt {
    int key;

    crypt(int k) {
        key = k;
    }

    void setKey(int k) {
        key = k;
    }

    int getKey() {
        return key;
    }

    char encryptChar(char c) {

```

```

        // encryption code
    }

    char decryptChar(char c) {
        // character decryption code
    }

    String encryptString(String s) {
        // string encryption code
    }

    String decryptString(String s) {
        // string decryption code
    }
}

```

In this version of `crypt`, the encryption key has been moved from a method parameter to a data member of the class, `key`. A constructor was added that accepts an encryption key when the object is created, along with access methods for getting and setting the key. The `public static` declarations for the `encrypt/decrypt` methods have also been removed, which requires you to have an instance of the `crypt` class to use the methods. This makes sense, because the class now has a data member (`key`) that affects the methods.

This revised design of the `crypt` class makes much more sense from a Java programming perspective. Of course, it won't run any faster or perform better encryption, but that's not the point. The point is that object-oriented design practices are a fundamental part of the Java language and should be followed whenever possible.

OPERATOR OVERLOADING

Operator overloading, which is considered a prominent feature in C++, is not supported in Java. Although roughly the same functionality can be implemented as methods in Java classes, the syntactic convenience of operator overloading is still missing. However, in defense of Java, operator overloading can sometimes get very tricky. The Java developers decided not to support operator overloading to keep the Java language as simple as possible.

Although operator overloading is a pretty useful feature in C++, its usage is highly dependent on the types of C++ classes with which you're dealing. For example, more fundamental C++ data structure classes, such as string classes, make heavy use of operator overloading, whereas others may not use it all. The amount of porting work you have in front of you depends on how much your C++ code relies on operator overloading.

The only way to convert overloaded C++ operators to Java is to create equivalent Java methods with the same functionality. Keep in mind that the Java methods will be called differently than the C++ overloaded operators. This means you have to dissect your code carefully to find out where each operator is used and then convert the code to a method call.

Let's look at an example; **Listing 9** contains a complex number class with overloaded operators.

Listing 9: The C++ Complex class with overloaded operators.

```
class Complex {
    float real;
    float imaginary;

    Complex(float r, float i);

    Complex operator+(const Complex& c) const {
        return Complex(real + c.real, imaginary + c.imaginary);
    }

    Complex operator-(const Complex& c) const {
        return Complex(real - c.real, imaginary - c.imaginary);
    }
};
```

The C++ Complex number class supports overloaded operators for addition and subtraction. The following is an example of how this class is used:

```
Complex c1(3.0, 4.0);
Complex c2(5.0, 2.5);
Complex c3 = c2 - c1;
```

The subtraction of the Complex objects is syntactically the same as subtracting two fundamental data types. However, this capability adds a significant amount of complexity to C++ that the Java architects wanted to avoid. So, although you can't provide the same syntactic approach in Java, you can provide methods with similar functionality. Listing 10 contains the Java version of the Complex class, complete with method versions of the overloaded operators.

Listing 10: The Java Complex class.

```
class Complex {
    float real;
    float imaginary;

    Complex(float r, float i) {
        real = r;
        imaginary = i;
    }

    Complex add(Complex c) {
        return (new Complex(real + c.real, imaginary + c.imaginary));
    }

    Complex subtract(Complex c) {
        return (new Complex(real - c.real, imaginary - c.imaginary));
    }
}
```

The most obvious change in the Java version of Complex is the renaming of the operator overloaded methods to add and subtract. The Java Complex class is used like this:

```
Complex c1 = new Complex(3.0, 4.0);
Complex c2 = new Complex(5.0, 2.5);
Complex c3 = c2.subtract(c1);
```

You can see how the subtraction operation isn't quite as intuitive using the Java approach. Nevertheless, it does work. Notice also that the Complex objects are created using the new

high quality - competitive rates - 16 years experience - award winning

Full Spectrum Software Development & Testing

Device Drivers
Porting Plug-ins
TCP/IP

Carbon / OSX

Cross Platform Development

One Bridge Street
Newton, MA 02458

617-965-0029

www.FullSpectrumSoftware.com

competitive rates - 16 years experience - award winning - high quality

Presto Vivace, Inc.

Fast and Lively Public Relations

Presto Vivace specializes in public relations for small technology companies. Our press contacts database is now available for companies to manage their own publicity.

For only \$99, you can use our professional database to place your press releases. Available in AppleWorks format; e-mail marshall@prestovivace.biz for sample.

4902 Powell Road, Fairfax, VA 22032

703/426-5876, fax 426-5892

<http://www.prestovivace.biz/>

operator. This is a result of the differences between memory management in Java and C++, which you learn about when you get into pointers a little later in this article.

AUTOMATIC COERCIONS

Automatic coercion refers to the implicit casting of data types that sometimes occurs in C and C++. For example, in C++ you are allowed to assign a `float` value to an `int` variable, which can result in a loss of information. Java does not support C++ style automatic coercions. In Java, if a coercion will result in a loss of data, you must always explicitly cast the data element to the new type.

The following is an example of an automatic coercion in C++:

```
float f = 3.1412;
int i = f;
```

Some C++ compilers may actually generate a warning for this code, but it's not considered an error. In Java, on the other hand, this code results in a compile error. It is easily fixed with an explicit cast, such as this:

```
float f = 3.1412;
int i = (int)f;
```

COMMAND-LINE ARGUMENTS

The command-line arguments passed from the system into a Java program differ in a couple of ways from the command-line arguments passed into a C++ program. First, the number of parameters passed differs between the two languages. In C and C++, the system passes two arguments to a program: `argc` and `argv`. `argc` specifies the number of arguments stored in `argv`. `argv` is a pointer to an array of character pointers containing the actual arguments. In Java, the system passes a single value to a program: `args`. `args` is an array of `String` objects that contains the command-line arguments.

In C and C++, the command-line arguments passed into a program include the name used to invoke the program. This name always appears as the first argument, and it is rarely used. In Java, you already know the name of the program because it is the same name as the class, so there is no need to pass this information as a command-line argument. Therefore, the Java runtime system passes only the arguments following the name that invoked the program.

Listing 11 contains a simple C++ program that prints out the argument list.

Listing 11. A C++ program to print the argument list.

```
#include <iostream.h>
#include <string.h>

void main(int argc, char* argv[])
{
    for(int i = 1; i < argc; i++)
        cout << argv[i] << " \n";
}
```

This program simply iterates through each argument using a `for` loop, printing each to the standard output stream. Notice that the loop starts at 1 to avoid printing the name of the program itself (the first argument). **Listing 12** contains the Java equivalent, the `ArgPrint` application class.

Listing 12. The Java `ArgPrint` class.

```
public class ArgPrint {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++)
            System.out.println(args[i]);
    }
}
```

The `ArgPrint` class contains one method, the `main` method. This is the Java equivalent of the C/C++ `main` function; it is called when the Java program is executed. Notice that the `main` method takes an array of `String` objects as parameters. The usage of this array highlights another interesting difference between Java and C++, arrays. All Java arrays have a data member called `length` that can be used to determine how many elements an array contains. In this case, `length` is used to iterate through the array of `String` arguments.

I/O STREAMS

You probably noticed the absence of the `cout` standard output stream object in the Java `ArgPrint` class. There's a good reason for this: Java doesn't implement any of the standard C++ stream objects. However, it does contain similar equivalents. The Java `System` class implements three stream objects that are very similar to the C++ standard stream objects: `in`, `out`, and `err`. You access these stream objects with the `System` class.

The `ArgPrint` class showed how to use the `out` stream object to output text. The `out` object is of type `OutputStream`, and it contains a number of methods, such as `println`, for outputting data to the standard output stream. Similarly, the `in` and `err` objects contain a variety of methods for performing stream input and output. In most cases, you can convert standard C++ stream I/O code to Java stream I/O code by simply changing the references from `cin` to `System.in`, and so forth. Because Java doesn't support operator overloading, you also need to change any `<<` or `>>` operations to the equivalent Java method calls.

STRINGS

C and C++ have no built-in support for text strings. The standard technique adopted among C and C++ programmers is that of using null-terminated arrays of characters to represent strings. In Java, strings are implemented as first class objects (`String` and `StringBuffer`), meaning that they are at the core of the Java language. Java's implementation of strings as objects provides several advantages:

ThinkfreeOffice

COMPATIBLE WITH MICROSOFT WORD, EXCEL AND POWERPOINT

WORDPROCESSOR • SPREADSHEET • PRESENTATION GRAPHICS

The Affordable Office Alternative!



Three High-Performance Applications

Thinkfree Write

Thinkfree Write is a powerful word processing application that enables you to create rich, professional quality documents and Web pages. You can insert tables, images, and clipart, or even apply custom layouts to your document...then effortlessly proofread your work with the easy-to-use spelling and auto-correction features.

Thinkfree Calc

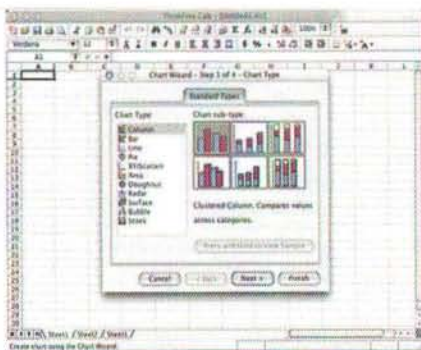
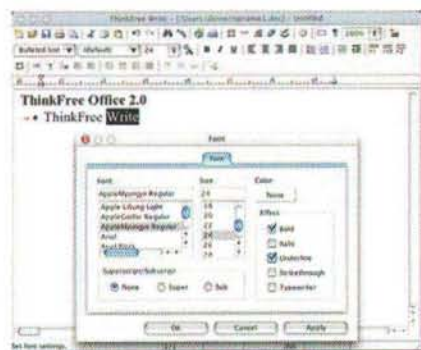
Thinkfree Calc is a full-featured, easy-to-use spreadsheet application that can easily tackle the most complex analytical tasks with over 40 charts and 300 function capabilities. Thinkfree Calc opens, edits, and saves directly into the Microsoft Excel (.xls) format, so users can seamlessly share documents and collaborate with Microsoft Office users.

Thinkfree Show

Thinkfree Show enables you to create high-impact presentations including animation effects, drawings, images, clipart, and other graphic features. Thinkfree Show opens, edits and saves directly into the Microsoft PowerPoint (.ppt) format.

CyberdrivePlus

A free, one-year subscription to CyberdrivePlus is also included. CyberdrivePlus provides you with secure, Internet file storage and free online software upgrades!



"Thinkfree is a best-of-breed program that will exceed your expectations."
— Jeffery Battersby



"Thinkfree Office is the next best thing and then some."
— Deborah Shadovitz



"Thinkfree Office is an impressive attempt to crack the seemingly impenetrable productivity market."
— Chris Ward

amazon.com

Apple Store



Apple Specialist



ONLY
\$49⁹⁵

- The manner in which you create strings and access the elements of strings is consistent across all strings on all systems.
- Because the Java string classes are defined as part of the Java language, and not part of some extraneous extension, Java strings function predictably every time.
- The Java string classes perform extensive runtime checking, which helps eliminate troublesome runtime errors.

Although it's easy to see the advantages of Java strings, converting C++ code to Java that makes heavy use of null-terminated character arrays is another issue. Next to pointers, string code is probably the most troublesome code to convert to Java. You simply must get dirty with the details and figure out exactly what is happening to be able to change the code to use Java string objects. The other problem is that character arrays are used extensively in almost every C/C++ program.

Listing 13 contains a simple C++ function that manipulates null-terminated character strings, **ReverseIt**.

Listing 13: The C++ ReverseIt function.

```
char* ReverseIt(const char* szText) {
    int len = strlen(szText);
    char* dest = new char[len];

    for (i = (len - 1); i >= 0; i--)
        dest[len - i - 1] = szText[i];
    return dest;
}
```

The **ReverseIt** function takes an array of characters and returns an array of characters that is the reverse of the original. It simply creates a new array and copies each character from the original array in reverse order. Notice, however, that there is nothing stopping you from writing code that overruns an array bound, or even from manipulating the character pointers. Even though this code works fine, the very nature of C++ makes it prone to potentially dangerous pitfalls implemented at the programmer's discretion. Take a look at the Java version of the same function, **reverseIt**, which is now a method in the **Reverse** class in **Listing 14**.

Listing 14: The Java Reverse class.

```
class Reverse {
    String reverseIt(String s) {
        int i, len = s.length();
        StringBuffer dest = new StringBuffer(len);

        for (i = (len - 1); i >= 0; i--)
            dest.append(s.charAt(i));
        return dest.toString();
    }
}
```

The Java **reverseIt** method has no mention or use of arrays. In Java, strings are first class citizens implemented by the **String** and **StringBuffer** classes; they are genuine data types that can be manipulated like integers or floating-point numbers. All modifications to Java strings must take place through the class methods defined in the

String and **StringBuffer** classes, as you can see in the **reverseIt** method.

POINTERS

Most developers agree that the misuse of pointers causes the majority of bugs in C/C++ programming. Put simply, when you have pointers, you have the ability to trash memory. C++ programmers regularly use complex pointer arithmetic to create and maintain dynamic data structures. In return, C++ programmers spend a lot of time hunting down complex bugs caused by their complex pointer arithmetic.

The Java language does not support pointers. Java provides similar functionality by making heavy use of references. Java passes all arrays and objects by reference. This approach prevents common errors due to pointer mismanagement. It also makes programming easier in a lot of ways, because the correct usage of pointers is easily misunderstood by all but the most seasoned programmers.

You may be thinking that the lack of pointers in Java will keep you from being able to implement many data structures, such as dynamically growable arrays. The reality is that any pointer task can be carried out just as easily and more reliably with objects and references. You then benefit from the security provided by the Java runtime system; it performs boundary checking on all array indexing operations.

Pointers are no doubt the most difficult aspect of moving C/C++ code to Java, because most C/C++ programs are riddled with pointer use. The first line of attack in converting pointer code is to convert all character arrays to Java strings. Once you do this, you'll probably be surprised at how much pointer-dependent code was weeded out.

The next phase of pointer conversion is object creation/destruction. In C++, the typical way objects are used is to create a pointer to an object variable and then use the **new** operator to create a new object and assign it to the variable. Once you finish with the object, you call **delete** on the pointer variable to clean up things. This procedure isn't all that different in Java; it's just missing the final step.

In Java, objects no longer being used are automatically cleaned up by the Java garbage collection system, which is typically implemented as a low-priority system thread. Because the Java system itself handles cleaning up unused objects, you don't have to worry about cleaning up after yourself. This may be your one opportunity to make a mess and not have to clean up after yourself, so take advantage of it! To better understand the differences between working with objects and pointers in C++ and Java, let's look at an example. **Listing 15** contains a C++ class with a member object pointer.

Listing 15: The C++ Rocket class.

```
class Rocket {
    Booster* booster = 0;

    Rocket() {
```

```

    booster = new Booster();
}

~Rocket() {
    if (booster != 0) {
        delete booster;
        booster = 0;
    }
}
};

```

The constructor for `Rocket` initializes the `booster` member variable by creating a new object. Then the destructor for `Rocket` cleans up the `booster` member by deleting it and setting it to 0. This is a painfully simple example, but it helps to learn things in small doses. **Listing 16** contains the Java version of this same class.

Listing 16: The Java Rocket class.

```

class Rocket {
    Booster booster;

    Rocket() {
        booster = new Booster();
    }
}

```

The Java code is much more simpler, even in this case where there is relatively little happening with the C++ pointers. Notice that the `booster` member variable isn't

initialized to 0 in the Java version of `Rocket`. This highlights a subtle feature in Java; all member variables are set to 0 or null if they are created without being initialized. Notice also the absence of a destructor in the Java code; the Java garbage collector takes care of cleaning up the `booster` object when it is no longer in use.

Note: Java actually supports a method very similar to a C++ destructor, the `finalize` method. The `finalize` method is called whenever an object is destroyed by the garbage collector. However, due to the nature of the garbage collector itself, Java does not guarantee that the `finalize` method will get called for all objects. In other words, don't rely on it getting called in your code!

If you're concerned about how Java gets by without using pointers, keep in mind that the Java system is certainly using pointers under the hood. The trick is that you are forced to do everything under the guise of references. This may seem like a pretty big limitation at first, but once you get in the habit of using references you'll see that you aren't really missing anything. Knowing this, another phase of porting C++ code to Java is converting pointers to references in the C++ code before you even start fooling with Java. Try converting some C++ code to being entirely reference-based and then take a stab at moving it to Java. You'll be in for a much smaller headache if you take this route.

ListSTAR®

www.liststar.com

The most flexible email processing system available. Easily create mailing lists or email-on-demand services. Use built in rules and/or AppleScript/AppleEvents to handle any email task, no matter how simple or complex. Demo available.

MacRADIUS™

www.macradius.com

The easiest to use RADIUS server available. The groups feature allows you to make changes for a large number of users in one easy step. Enabling or disabling access for a user or a group of users is a one click operation, without having to stop the server. Support for AppleScript/AppleEvents makes it easy to control MacRADIUS. Demo available.

SimpleText Filter

Plug-in for EIMS*

www.mcfsoftware.com/stf/

Easy to use header and content filtering. Scan incoming messages for sequences of text, digits, etc. Base64 messages are decoded so you can check for content despite attempts to hide the text. Demo available.

Auto Reply

Plug-in for EIMS*

www.mcfsoftware.com/ar/

This plug-in allows you to easily set up auto-reply messages for users. Addresses that have been sent auto-reply messages are tracked, preventing auto-reply message loops. Demo available.

Address List Sorter

www.mcfsoftware.com/als/

Fast and powerful utility for sorting and cleaning email lists. Sort email address lists in alphabetical or domain order, remove duplicates and improperly formed addresses. Demo available.

*EIMS—Eudora Internet Mail Server (www.eudora.co.nz)



...*simply*
dependably
engineered

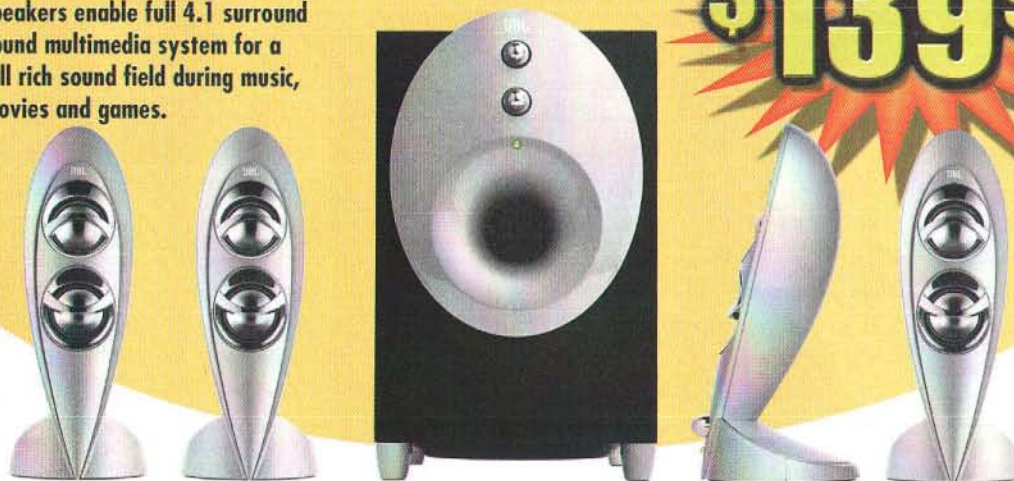
www.mcfsoftware.com

Macintosh

**Bring your games, MP3's, or DVD's to Life
with Professional Sound on Macintosh!**

JBL Invader Speakers 4.1 Surround Sound!

The latest in sound innovation, these speakers enable full 4.1 surround sound multimedia system for a full rich sound field during music, movies and games.

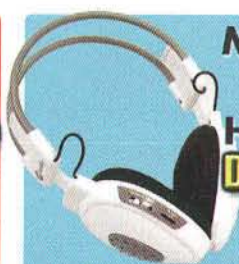


\$139⁹⁵

Griffin PowerWave!

**DevDepot Price
\$94⁹⁵**

A powerful and extremely flexible computer audio tool, PowerWave can record any mic or line input into your computer. Use it as an external amp to connect any set of home speakers to your computer!



Macally Noise Reduction Head Phones!

**DevDepot Price
\$64⁹⁵**

The perfect companion for a long flight, iPod, or anyone who wants the best listening experience! Cancels out background sound! Includes an air line adapter to plug directly into most airplanes.

**Now
for New
iPods!**

iTrip for iPods

**DevDepot Price
\$34⁹⁵**



Let's you play your music through any FM radio -- in your car, at a party, wherever! Needs no batteries! Programmed from your iPod and now available in two versions, for all iPods!

Sound Machine!

Harman Kardon SoundSticks USB Speakers!

Harman Kardon's SoundSticks-a 3-piece speaker system designed for easy plug-and-play. SoundSticks was created for USB-equipped Apple computers, such as iBook, iMac, PowerBook, Power Macintosh G3, and Power Mac G4.

\$159⁹⁵



JBL Creature Speakers!

Ideal for iPod and iMac. A 3 piece speaker system including two desktop speakers and a subwoofer. Both speakers and the subwoofer are magnetically shielded to provide the greatest protection against image distortion on a computer monitor

\$99⁹⁵



KEYSPAN Digital Remote!

DevDepot Price
\$34⁹⁵



A remote control for your USB Mac! Use your PowerBook as a DVD player from the couch, great for presentations!

DEV DEPOT[®]

www.devdepot.com

877-DEPOT-NOW

MULTIPLE INHERITANCE

Multiple inheritance is a feature of C++ that enables you to derive a class from multiple parent classes. Although multiple inheritance is indeed powerful, it is complicated to use correctly and causes lots of problems otherwise. It is also very complicated to implement from the compiler perspective.

Java takes the high road and provides no direct support for multiple inheritance. You can implement functionality similar to multiple inheritance by using interfaces in Java. Java interfaces provide object method descriptions, but contain no implementations. Let's take a look at an example of C++ code that uses multiple inheritance:

```
class InputDevice : public Clickable, public Draggable {  
    // class definition  
};
```

The C++ `InputDevice` class is derived both from the `Clickable` and `Draggable` classes. This means that `InputDevice` inherits all the data members and methods implemented in both of these classes. The closest you can get to this in Java is to make `Clickable` and `Draggable` interfaces, which can contain method definitions but no actual method code or data members. The `InputDevice` class can implement these interfaces using the `implements` keyword:

```
class InputDevice implements Clickable, Draggable {  
    // class definition  
}
```

As you can see, you may have your work cut out for you if you are trying to move C++ code to Java that relies on lots of multiply inherited classes. Even so, the Java interface approach is not all that bad; you just have to juggle the actual method bodies and possibly implement more derived classes to contain them. Nevertheless, the primary goal of uniting separate logical organizations into a more derived one will still be attained.

INHERITANCE SYNTAX

The multiple inheritance example brings up another important difference between C++ and Java: inheritance syntax. In C++, you specify inheritance by using a colon after the newly derived class, followed by the parent class or classes:

```
class B : public A {  
    // class definition  
};
```

In Java, the `extends` and `implements` keywords are used to indicate inheritance:

```
class B extends A {  
    // class definition  
}
```

Fortunately, this change can be made in your C++ code as a simple search and replace, for the most part. The only

hangup will be changing the syntax for classes using multiple inheritance, in which case you have to do some real work anyway

ACCESS MODIFIERS

Access modifiers are supported in both C++ and Java, but the methods of declaring them are different in each. In C++, you declare access modifiers as a label above a group of class members:

```
class A {  
    public:  
        int x, y;  
    private:  
        float v;  
};
```

In Java, you can do the same thing, but you apply the access modifiers a little differently. You set them for each individual declaration:

```
class A {  
    public int x, y;  
    private float v;  
}
```

In this way, Java access modifiers aren't labels at all; they really are modifiers. Converting access modifiers in C++ code is pretty simple; just go through and remove each label, adding the appropriate modifier to each variable declaration or method following the original label.

FRIENDS AND PACKAGES

Java has no friends! To prove it to you, Java doesn't have any concept of a friend class, whereas C++ does. A *friend* class in C++ is one that has access to all the data and methods in another class, regardless of the visibility of the member data. Java has no exact equivalent to the friend concept, but it does have an access type that enables a specific group of classes access to member data, which is somewhat similar to the friend idea.

This Java access type is often referred to as the *default* type, because there is no keyword used to apply it. You may also see it referred to as the *friendly* access modifier. It has the effect of giving classes in the same package as the class in question access to the member variable or method declared with default access. To give a member variable or method default access, you simply don't use an access modifier, like this:

```
class A {  
    public int k;  
    private int h;  
    int i, j;  
}
```

In this example code, `k` and `h` take on the specific access modifiers they are declared with, and `i` and `j` assume default, or package, access. To convert friendly C++ code to Java, you should be able to combine friend classes together in the

same package and declare many of their shared members with default access.

BOOLEANS

In C++, there is no real boolean type; boolean values (true and false) are usually implemented as integers, with zero being interpreted as false and nonzero being interpreted as true. This usage resulted in somewhat of a half-hearted attempt by C/C++ programmers to use booleans in their code. Take a look at this example:

```
if (--numLives)
    isDead = 1;
```

There are a couple of assumptions being made here that don't sit well from a purist programming perspective. First, if statements should always be based on a boolean value. However, as this example shows, programmers often exploit the fact that C/C++ if statements (and other conditional statements) operate on values either being zero or nonzero, rather than true or false. The `isDead` variable, which clearly has a boolean meaning, uses 1 to represent true.

Unlike C/C++, Java supports boolean values as a type all their own. As a matter of fact, this example wouldn't work in Java because Java if statements fully expect boolean types, not integers parading as booleans. The Java version would look like this:

```
if (--numLives <= 0)
    isDead = true;
```

The if statement has been changed to result in a boolean outcome. This outcome was implied in the C++ code, but it is made clearer and more consistent in the Java version. The `isDead` variable has been changed to a boolean type. Notice that `true` and `false` are legitimate Java keywords (as is `null`).

Most C/C++ code contains this "integer as boolean" approach and will therefore have to be fixed in a Java port. However, these changes are pretty simple and shouldn't take too much time.

SUMMARY

In this article, you learned about the different challenges you are faced with when porting C and C++ code to Java. Even though Java is a close cousin to C++, you saw how there are enough differences to make any conversion process a decent amount of work. On the other hand, the final benefits of moving your existing code base to Java can far outweigh the potential difficulties you encounter while moving it.

The goal of this article isn't to bog you down with hundreds of detailed examples of complex code conversions, but rather to highlight the primary problem areas where the bulk of the code conversion will take place. If you take it a step at a time, you'll find that moving C/C++ code to Java isn't all that hard. (If it makes you feel any better, I was able to convert a complex set of sprite classes from C++ to Java in a couple of days.) Just be patient and think about how cool your code will be running live on the Web!



Valentina

Object-Relational SQL Database

**The fastest database engine
for MacOS/Windows**

**It operates 100's and sometimes
a 1000 times faster than other systems**

www.paradigmasoft.com

Hosted by macserve.net
Download full featured evaluation version

By Michael R. Harvey, Reviews Columnist

Mac OS X training from TackyShirt

Training that doesn't suck, really.

Their tag line is, "because training and fun are not mutually exclusive." For most training materials out there, however, that is not the case. Then came Sam Crutsinger, President and "Media Kingpin" of TackyShirt Inc., a company he formed to produce quality technical education that is also entertaining. Says Crutsinger, "This is the 21st century. We should be able to present education in a way that keeps people engaged and gets the point across. I hate to read. [David] Pogue's books are great, but I still have to sit down and read the thing to absorb it." Add to that his many years of experience in video editing and production, and a desire to do it different, and better than anyone else, and you have the ingredients for an interesting recipe.

The core idea behind the training was to make it educational and fun. In other words, don't bore the viewer into a coma. So, to start, they chose to keep each of the segments under fifteen minutes (based on educational research published in a study from NYU). Next find some talent. Crutsinger called on some of the most well known, as well as some of the most knowledgeable, folks in the Mac community. Andy Ihnatko, long time columnist for Macworld magazine, Bo LeVitus, author of many Mac related books, Shawn King, host and Executive Producer of Your Mac Life (an internet based radio show), and John Welch, a regular presenter at MacWorld Expos, and columnist for MacTech magazine. He got these four names together, put them in front of sets not seen before in any technical training, and for two weeks committed to tape pretty much everything they uttered about the Mac, and Mac OS X. And, it seems, some things not quite Mac related (there are a few Easter Eggs on the DVD, one of which is outtakes of some behind the scenes shenanigans). Then came the hard part, editing that mess down to usable, and useful, nuggets. A daunting task when you take into account how much detail went into every segment.

The first fruit of this labor is entitled Mac OS X Disc 1: The Basics. Yeah, pretty dull title. Fortunately, the dullness ends there. On the disc is nearly four hours of fun education. In rotating pairs, Shawn, John, Andy, and Bob take turns presenting some aspect of Mac OS X to the new user. They are accompanied by picture in picture computer screen



shots, animations, and active desktop screens that show exactly the what, where, and how of the topic they are discussing. And it is a discussion. They are not lecturing to an invisible audience, but rather seem to be carrying on a conversation with each other, and the viewer. Of course, there are other characters that constantly attempt to steal the spot light including chatting set pieces, and icons run amok, to name a few. They do not, however, distract from the learning. They actually seem to enhance it.

This is a basics learning tool, though. What can advanced users get out of it? Well, we watched it. We thought it would bore us to death. This is, after all, the basics. We, and you, are almost certainly well beyond that. Surprisingly, there are a few tidbits of knowledge to be had. There is always that one little thing you never noticed before that will help make you more productive, and this training may just show it to you. Putting that aside for the moment, this product is aimed at those who are starting from square one. So, we showed it to a couple of new Mac OS X users. One who was completely new to Macs, and a switcher from OS 9. Our newbie enjoyed it immensely,

and got a lot out of the DVD. He felt though, that some knowledge was presumed in the material, and said if he had some experience with OS 9, he would have gotten more from the training. Our OS 9 upgrade liked it quite a bit, too. Her main complaint was that Andy Ihnatko talked too fast, and lost her at some points.

Our next question was how can this material be useful to MacTech readers? By in large, we figure the folks reading this review would not get their investment out of it. However, every one of us knows at least one person that really, *really*, needs this DVD. Maybe that friend, colleague, family member will stop calling every other day if they watch it. For some of us, this training could help at the job. Consultants, and administrators with clients and users in need of updated training will all find Tackyshirts material invaluable.

All is not lost for the advanced user, however. This is Disc 1 of what will ultimately be a four disc set. No date has been set for subsequent releases, but follow on discs will cover topics such as troubleshooting, and backup strategies (Disc 2), intermediate level training, and popular applications (Disc 3). Disc 4 is where most advanced users will get the real goods. This disc is scheduled to cover under-the-hood kinds of topics, teaching the Unix underpinnings, and advanced capabilities of Mac OS X.

TackyShirt Presents Mac OS X Disc 1: The Basics is currently available only from the company at their web site. Disc 1 is \$39.99, plus \$4.95 shipping per item (international shipping is \$9.90 per item). Future installments are slated to be the same price. You can even get yourself one of their t-shirts sporting the cool TackyShirt logo for \$10.85 (one buck more for us XXL types).

One final warning. When you do sit down to watch this, don't have any other plans. You may think you're just going to watch one segment real quick, but you are just lying to

yourself. This stuff is digital crack, and before you realize it, you'll have lost two hours to it. Don't have a wedding or a funeral to go to, because you will be late, and end up cheesing off the others, especially if you are the guest of honor at either of those events.

www.tackyshirt.com

Whistle Blower

Enterprise server monitor and restart utility
whistleblower.sentman.com

Connect to and validate the response from web servers, cgi scripts and over 23 other types of servers.

Send email, pages and perform unattended restarts via MasterSwitch or PowerKey.

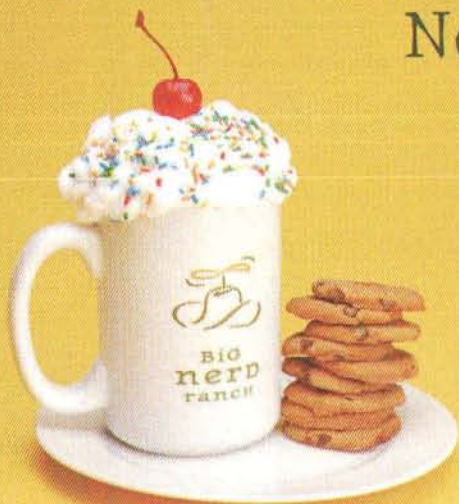
Shifts make sure that the person on call when the server goes down is the one who gets the page.

68k, PPC and Carbon

Web based administration lets you check on and restart your servers from anywhere.

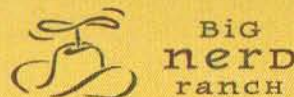
Customize your response to an outage with Apple Script.

email us at whistleblower@sentman.com



Now serving Cocoa[®] just the way you want it.

Training for Mac OS X doesn't have to be the same old flavor. Reserve your seat in a class at our scenic lodge location, or have experts come to you for **Extreme Mentoring**. Two weeks of on-site instruction and collaboration, customized to the requirements of your project. Book now for 2003. See why we're different.



Intensive Classes for Programmers
www.bignerdranch.com

By Ron Davis

Book Review: Chris Crawford On Game Design

I first picked up *Chris Crawford On Game Design* at the local bookstore because I am a wannabe game programmer and often look at game design books. I kept picking it up because there are a lot of Mac screen shots and descriptions of Mac games in it. This is very unusual for a game book, but Chris Crawford is a game programmer from way back.

One thing to know about this book is it is very much about Chris Crawford. Very oriented toward his opinion and his philosophy, with nary a line of code in the whole book. The book is broken up into two sections; the first part is an overview of game design and the second is a history of all of the games Chris has written. Because of this his personality comes through very strongly. Maybe it is just hard to write about the cool things in the code you've written and not come off seeming an egomaniac, but after reading the second half of the book I didn't really like Chris. Now that I've read the first half, I'd say he knows his stuff, but I probably wouldn't want to hang with him, as he might feel the need to point out my flaws.

Before you get a bad opinion of me, let me quote the beginning of the chapter entitled *Random Sour Observations*:

"You would never guess it from my comments in this book, but I have a reputation for, shall we say, outspokenness. That reputation is mostly on the mark, although it is often colored by the anger of those whom I have skewered. My particular talent is not for detecting problems – anybody can bitch – but rather for phrasing my criticisms in a style that hits hard. I hold euphemism and tactful ellipticity in contempt; integrity demands the expression of truth in the clearest and most compelling terms."

The tone of the book is biting. He freely lambasts everyone in computer games with a broad brush. So if you have a thin skin and don't want to hear someone say all programmers are autistic, lack all social skills, and will therefore never be able to create a game that reaches anyone but horny, violent young men, don't buy the book.

Now on to the good stuff. The first half of the book Chris talks about the history of games, both computer and otherwise,

the core concepts of Play, the requirements of Challenge, Conflict and Interactivity in computer games. Then he goes on to discuss the missing element of creativity in modern computer games, and common mistakes game programmers, companies, and designers keep making. There is a chapter dedicated to what he thinks a game designer needs to know entitled *The Education of a Game Designer* and one that lists a bunch of games he'd like to write. Then he talks about Storytelling and how it is lacking in modern games and people don't even seem to know it. The last chapter in this section of the book is the previously mentioned chapter of sour observations on the gaming industry as a whole.

There is some great stuff in these chapters, and rather than go through them one at a time, I'm just going to talk a little about some of the stuff I thought was cool.

In the chapter on challenge there is a long and interesting discussion of how the brain does things and how it learns. This he closely ties to game play and how the complexity of a game can increase without losing the player. Whenever our brains learn how to do a complex task we first have to think of each little step, and this thinking is slow. As we repeat the steps we shove the doing of the steps down in to our cerebellum and no longer have to think about things to do them. When playing a game we do this as we learn the game. So stuff that was slow and complex at the beginning isn't even thought about at the end. On the other hand, in games that are sequels, you either have to make the experienced player redo the now easy stuff or lose the new player with the overwhelming complexity of the game.

The chapter on Interactivity is the core of his philosophy of computer games. It is interactivity that makes computer games different from other games. Its really broader than that. Interactivity makes computers in general different. You can type things on a typewriter, but it is the interactivity, the ability to react to mistakes and change them, that make a word processor more useful. Chris points out good computer games are interactive. Unfortunately many modern games have ceased to be interactive today.

When Chris says creativity is missing from computer games today he's talking about a couple of things. First nothing new is really happening. People are looking at the kind of game they

want to write first and then making up some half assed story to do the same thing previous versions of this type of game did. And he is right. Is there really a difference between what the player did in Doom and what they do in Unreal? You run around and shoot things. They may look prettier. You may have new weapons, but really you are still running around shooting things.

Also missing from creativity is an understanding of "art" in general in game design. In his chapter on the education of a game designer, he talks about the lack of liberal arts education in game designers. For the most part game designers are programmers, and programmers are a lot more interested in the challenges of creating the game technically, the algorithms, the graphic engine, etc. than in the challenges the player faces. He gives a long list of books you should read if you want to be a game designer. The list is sure to leave you feeling like you are completely unread. There are only 4 computer books in the bunch, *Code Complete*, *The Mythical Man-Month*, *Algorithms* and *The Art of Computer Programming*, all of which are the basic texts every programmer should read. The other lists include everything from *The Way Things Work*, to *The Story of Law*, to *Walden* by Henry Thoreau, to Shakespeare, the *Federalist Papers*, and the New Testament.

I did like his suggestion in this chapter to "take up a mildly dangerous hobby" like motorcycling or sky diving. Now I can tell my wife it is job related.

The last half of the book is about individual games he wrote. It is an interesting history, written a lot like those

conversations you always end up in when hanging with other programmers who have been doing this a while. Talking about the challenges of a particular project, how they overcame them. You learn what he learned from each game and what he thinks did and didn't work. You also get a fascinating insight into the history of computer games. His first game was written on a computer that had no display. All input and output was done through a typewriter. Yet he wrote a tank battle game for it.

Chris worked for Atari and wrote a number of games for them. Then when he left Atari and had time on his hands, he bought a Lisa and started programming for the Mac. He wrote a number of Mac games and talks about them in the book. You can even go to his website (<http://www.erasmatazz.com/>) and download a number of them.

SUMMARY

Overall there is a lot to learn from this book. Not in the "How do I make 3D objects?" way, but in the how should this specific kind of computer program, a game, work for the user. It is about being a game designer, not a programmer. About creating all the stuff you do before you write a line of code. That's what makes it worthy of your bookshelf. If you want a programming book wait for my next column; it'll be on a more technical book.

It's 3AM on Sunday morning...

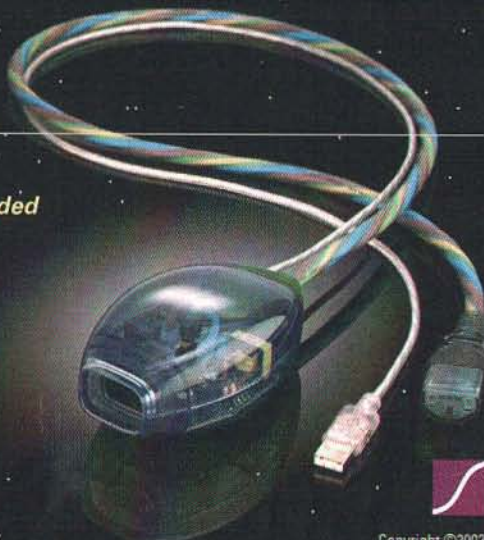
IS THE SERVER STILL RUNNING?

*End the
worry with*

Kick-off!

*The simple reliability
solution for Mac OS 9 and X*

- + **System-level Crash Detection**
patented hardware-software integration
- + **Automatic Crash Recovery**
system restart or power cycle, as needed
- + **Monitor Custom Applications**
software developer's kit and plug-ins included
- + **Scheduled Restarts**
cures memory leaks and fragmentation
- + **Restart after Power Failure**
even if shut down by your UPS
- + **Comprehensive Error Logging**
to help you track down problems
- + **Simple Installation**
just plug in AC cord and USB cable



For all these features *plus* six power outlets controllable by phone tones, schedules and scripts, consider our *PowerKey Pro 650 Admin*.

www.sophisticated.com

SOPHISTICATED CIRCUITS INC.

Copyright ©2002 Sophisticated Circuits, Inc. Kick-off! and PowerKey are registered trademarks of Sophisticated Circuits, Inc. Mac OS is a registered trademark of Apple Computer, Inc.

Available at
DEPOT
www.dewdepot.com

By John C. Daub

ARTIS Screen Tools

Useful utilities for pixel measurements.

A GREAT FIND!

Between my day job as a software engineer and occasionally fiddling with my hsoi.com web site in the evenings, I find myself every so often being concerned with pixels. I want to know how many pixels long is this? How many pixels are there between this and that? Will that many pixels fit onto the screen? What color is that pixel? Are all these pixels lined up? In trying to have my software adhere to Apple's Human Interface Guidelines and to have my web site be usable by any web browser, I need to employ tools to help me count pixels because my eyesight is too poor to count manually. ☺

Interface Builder has the terrific functionality of supporting guides and snapping-to when laying out your GUI, and the added cool factor having support for adhering to the Aqua Interface Guidelines. When designing in Interface Builder, I always have this support on because it generally alleviates the need to count pixels for relative measurements, e.g. is the widget 20 pixels from the left and right edges of the window, 14 pixels of vertical spacing, etc. While Interface Builder's support is useful, there are other measurements that sometimes need to be taken. For years I used a tool called colorScope, which served my "pixel" needs quite well. However, colorScope didn't make it over to Mac OS X. I dug around the 'net looking for a replacement tool. I found the ARTIS Screen Tools, a collection of four simple utility applications aimed towards solving a specific screen issue: measuring, ensuring fit, magnification, and ensuring alignment.

ARTIS Screen Tools is an application suite consisting of: ARTIS Screen Ruler, the screen ruler for your desktop; ARTIS Small Screen, small screen sizes on your desktop; ARTIS Screen Loupe, the magnifying glass for your desktop; and ARTIS Screen Guides, visual guide lines on your desktop.

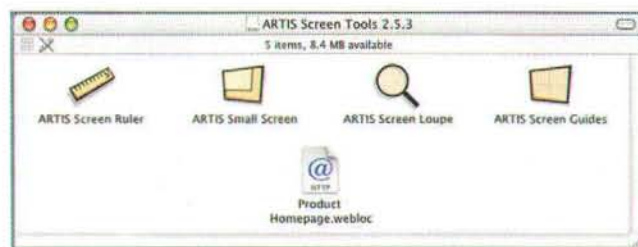


Figure 1. Contents of ARTIS Screen Tools 2.5.3.

THE TOOLS

Screen Ruler's purpose is measuring. You are given both horizontal and vertical rulers by which you can measure items on screen. The rulers float above all other windows, allowing control over their opacity, as well as control over the units and length of the ruler. Screen Ruler allows for quick and easy measuring of screen items, and is as simple to use as a physical ruler.

Small Screen's purpose is to help you ensure your window and its contents fit on screen. As developers we tend to be spoiled with a lot of screen real estate, but many of our users are still using small 640x480 screens. Small Screen places the thin black outline of a rectangle on screen wired to the sizes of well-established screen sizes, e.g. 640x480, 1024x768, or you can establish your own sizes. When Small Screen places the boundary on screen it can adjust for items such as the menubar so you can truly see how your window will fit and display on a monitor of that size. Small Screen also has a Browser Content mode which further adjusts the boundary rectangle to consider browser window content such as toolbars and scrollbars so you can see how the more popular browsers will squeeze your content and force it to reflow. Handy for debugging your code prior to posting it to your web site.

Screen Loupe is a magnification tool, like Apple's Pixie but more powerful. With Screen Loupe you can view pixels close up, up to 8x magnification. You can also view information about the pixel under the cursor: global coordinates, RGB color, HSB color, and HTML code color. I remember using this sort of tool when I was working on the Grayscale implementations of the PowerPlant Appearance Classes. I had to determine the correct patterns for an indeterminate progress bar and chasing arrows so

John C. Daub lives in Austin, Texas USA with his wife, three children, and all the BBQ he can eat. You can contact John at hsoi@hsoi.com.

LGAProgressBarImp and LGACHasingArrowsImp could reproduce them. Being able to zoom in on the pixel patterns of the Appearance Manager-generated widgets and pick up the RGB color values was a necessary part of recreating those widgets.

Screen Guides is a way to add guides when you can't have guide support or wish to augment existing guide support. Guides are lines that help you position your layout but don't appear in the final layout; they are tools to aid in supporting proper alignment of your content. You begin working in the Screen Guides application, creating and moving guides as necessary for your layout purposes. When you switch out of the Screen Guides application, the guides are no longer editable and you can begin your work. The guides float above your windows, so they'll be just where you'll find them useful while you work on your layout, but they won't get in your way.

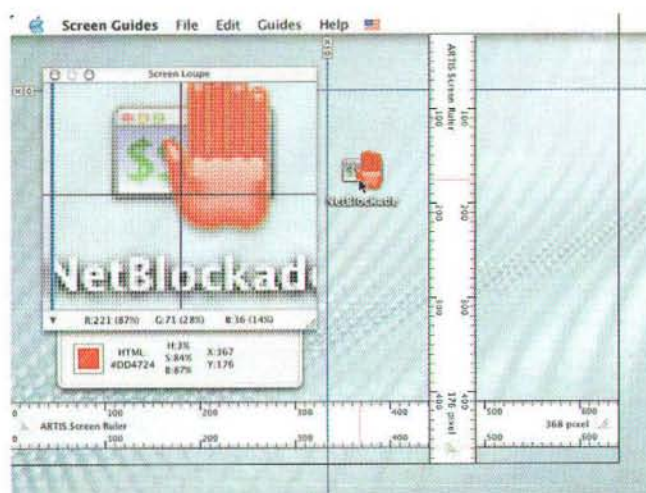


Figure 2: ARTIS Screen Tools at work.

The ARTIS Screen Tools are offered as shareware, US\$9.95 at the time of this writing, for the suite of all four tools. Once you register, functionality is unlocked in the applications: in Screen Ruler, the ability to use the vertical ruler; in Small Screen, the Browser Content sizes; and in Screen Guides the ability to use vertical guides. When I contacted the author for support, the response was quick, courteous, and helpful.

The ARTIS Screen Tools work together as a nice suite for handling pixel-measuring needs. Each application is lightweight and focused in its purpose, and serve to address needs that both software and web developers can encounter in designing, testing, debugging, and development of a successful user interface. If you find yourself needing to measure, fit, magnify, or align pixels, you'll find ARTIS Screen Tools a useful addition to your toolkit.

REFERENCES AND URL'S

ARTIS Screen Tools – <http://www.artissoftware.com/screentools/>

Apple Human Interface Guidelines –

<http://developer.apple.com/techpubs/macosx/Essentials/AquaHIGuidelines/index.html>

New PrimeBase 4.2 Replication Server

Check out the fully programmable Replication Server

- Bidirectional Updates supported
- Update 3rd-party DBMS
- Send Emails
- Post/get Data to/from Websites

**SQL-Runtime Plugin
for REALbasic
\$ 499,-**

All PrimeBase Server Software

- SQL-Runtime Libraries available
- SQL Database Server
- Application Server
- Replication Server
- Open Server

Available on the most popular platforms

- Completely cross-platform
- Full-text searching and indexing
- Mac OS & OS X
- Linux
- Solaris
- IBM AIX
- all Windows platforms

PRIMEBASE

SNAP Innovation GmbH
Altonaer Poststraße 9a
D-22767 Hamburg / Germany
www.primebase.com

e-mail: info@primebase.com

Fon: ++49 (40) 389 044-0

Fax: ++49 (40) 389 044-44

By Kevin Hemenway, Omnipotent Yodeler

Enabling CGI Scripts

Prepare your server for more powerful dynamic capabilities

Last month, we started down the road of adding dynamic content to our web server with the use of Server Side Includes (SSI). Often regarded as “simple” compared to languages like Perl and PHP, they can provide some quick turnaround to minor features like small single-item databases, changing content based on user whim or URL, and so forth. Regardless of whether you plan on using SSIs or not, we laid some important groundwork with our understanding of how modules work, Apache’s “block” configuration format, and a soft introduction to GET and POST.

All relatively Lego. Let’s break out the Technix.

A QUICK ‘HOW DO YA DO?’ TO CGI

SSI’s are built into Apache through the use of a module—when they’re enabled, the web server handles the request of a resource, processes the SSI statements within, and then sends the final output back to the user. Since Apache is a web server and not a programming language, the capabilities of the built-in SSIs are minimal—they’re not intended to be a full-fledged coding environment, and they never will be.

This is where CGI comes in. Meaning “Common Gateway Interface”, it defines a way for a web server to interact with external programs: Apache handles the request, sends some relevant information to the requested resource, and trusts the resource to handle the rest (including sending back the final content). If the resource doesn’t respond properly (either due to bad programming, incorrect permissions, etc.), Apache generates an error.

What does this really mean? Ultimately, it allows you to use any shell programming language to “do stuff” before showing the results to your visitors. Whether this means displaying content from a database, resetting passwords, saving a comment to a weblog, etc., as long as your program finishes properly, Apache doesn’t give a darn.

Most commonly, these “CGI scripts” are written in Perl, but you can use PHP, C, Bash, Ruby, Python, and anything else you may be interested in. Likewise, you can use them all in tandem—you’re not locked into any one programming language at any one time. Be forewarned, however: you *are* locked into security concerns: anything possible in your programming language of choice is doable within your CGI script, including deletion of files, exhaustion of memory, buffer overflows, poor security, and impolite use of resources. Add in the fact that all of these errors in judgment become publicly run-able by anyone who accesses your server, and you’ve got a whole mess of potential trouble.

ENABLING THE GATEWAY TO YOUR SCRIPTS

Learning about CGI follows my oft-repeated method of “search the `httpd.conf` for the feature you want”. So, open up `/etc/httpd/httpd.conf` in your favorite text editor and search for the word “CGI”—our first two matches look vaguely familiar:

```
LoadModule cgi_module          libexec/httpd/mod_cgi.so
AddModule mod_cgi.c
```

If you recall from our previous articles, Apache contains most of its features broken up into modules—the equivalent of plugins. To enable a module, two lines must exist uncommented (not preceded by a # character) in the `httpd.conf` file, an `AddModule` and a `LoadModule`. As previously with SSI, so too with CGI: our two lines already exist, and are already enabled. Let’s move on to our next search result, which is also similar to what we saw last month:

```
<Directory "/Library/WebServer/Documents">
# This may also be "None", "All", or any combination of
# "Indexes", "Includes", "FollowSymLinks", "ExecCGI",
# or "MultiViews".
Options Indexes FollowSymLinks MultiViews

AllowOverride None
Order allow,deny
Allow from all
</Directory>
```

Kevin Hemenway, coauthor of *Mac OS X Hacks*, is better known as Morbus Iff, the creator of disobey.com, which bills itself as “content for the discontented.” Publisher and developer of more home cooking than you could ever imagine (like the popular open-sourced aggregator [AmphetaDesk](http://AmphetaDesk.com), the best-kept gaming secret Gamegrene.com, articles for Apple’s Internet Developer and the O'Reilly Network, etc.), he’s been spending the last two months listening to every song in his iTunes library twice. Contact him at morbus@disobey.com.

Whereas last article the above result was triggered by the word `Includes`, it's `ExecCGI` this time around. Unfortunately, explaining `ExecCGI` without some prior knowledge will cause a little bit of confusion, so let's skip ahead to our next few matches before we go much further. I've truncated them for relevance:

```
# ScriptAlias: This controls which directories
# contain server scripts. ScriptAliases are essentially
# the same as Aliases, except that documents in the
# realname directory are treated as applications and
# run by the server when requested rather than as
# documents sent to the client.
#
ScriptAlias /cgi-bin/ "/Library/WebServer/CGI-Executables/"

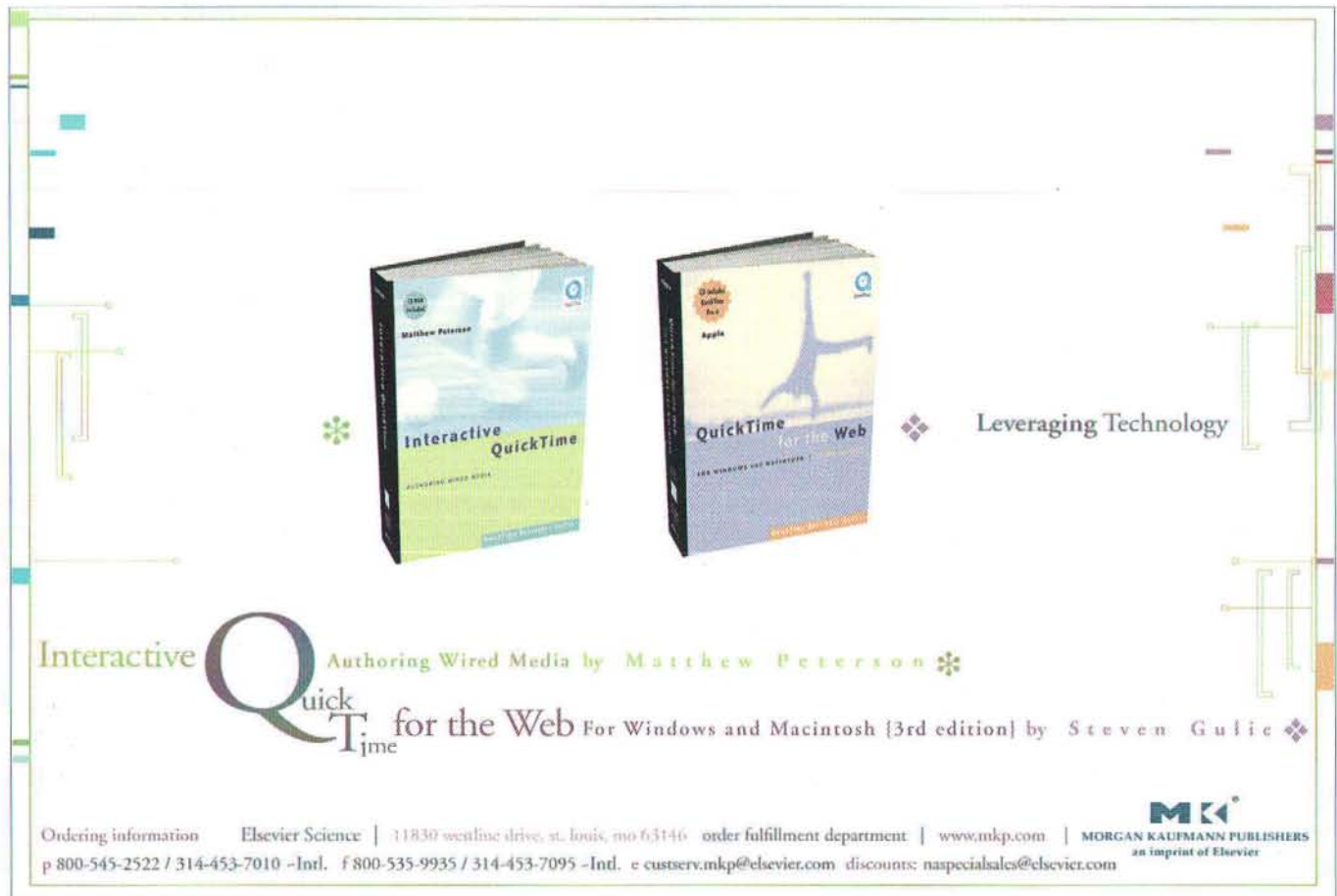
# "/Library/WebServer/CGI-Executables" should be changed
# to whatever your ScriptAliased CGI directory exists,
# if you have that configured.
#
<Directory "/Library/WebServer/CGI-Executables">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

The most common CGI configuration is all about segregation: "only files in *this* directory should be executed

as programs—everything else, anywhere else, are just plain old files that should be processed normally." From the standpoint of an overbearing system administrator, this makes sense. With only one directory that can execute code unconditionally, and only one person (you) who can put programs there, it can bring a little sanity to your day-to-day security paranoia.

This is how Apache is configured by default, and the above `httpd.conf` snippet shows these settings. Our first directive, `ScriptAlias`, lets us pick and choose which directory we'd like CGI scripts to live in. Anything within that directory, CGI script or not, will be executed by the web server as if it were a program. If something goes wrong, Apache will return an "Internal Server Error" to the requesting user-agent (ie. your visitor's browser).

`ScriptAlias` consists of two space-separated parts: the fake name and the real name. The fake name, `/cgi-bin/`, defines what URL will be used to access the real name, `/Library/WebServer/CGI-Executables/`. In the currently configured settings, whenever we access a URL like <http://127.0.0.1/cgi-bin/printenv>, we'll really be accessing the matching file stored at `/Library/WebServer/CGI-Executables/printenv`.



Interactive QuickTime

Authoring Wired Media by Matthew Peterson

QuickTime for the Web

For Windows and Macintosh [3rd edition] by Steven Gulic

Leveraging Technology

Ordering information Elsevier Science | 11830 westline drive, st. louis, mo 63146 order fulfillment department | www.mkp.com | MORGAN KAUFMANN PUBLISHERS
p 800-545-2522 / 314-453-7010 -Intl. f 800-535-9935 / 314-453-7095 -Intl. e custserv.mkp@elsevier.com discounts: naspecialsales@elsevier.com an imprint of Elsevier

The second part of our configuration is a block directive focusing on that specific location of our hard drive. It merely ensures that said directory has no special privileges besides that `ScriptAlias`: it can't override anything in the Apache configuration (`AllowOverride None`), and there are no special capabilities (`Options None`) associated with it.

At this point, if you only care about the directory segregation of CGI, you can skip over to our next subheading, "Let's See One of These CGI Thingies In Action!" If, on the other hand, you'd like to hear me get high-and-mighty over the "right" way of doing things, read on.

CGI EVERYWHERE: "A BRILLIANT PSYCHOLOGICAL THRILLER!"

In the very first "Untangling the Web", I professed a fondness and desire to teach you the "right" way to create URLs (*MacTech Magazine*, June 2003, "Untangling The Web", "Familiarity That Breeds An Uh-Oh!"). A good URL will remain in existence until the dusk of time, never needing to be changed, never needing to be refactored. It's not a snap-decision: if you find yourself changing your URLs around, you designed your site poorly (from an engineering, not visual, standpoint).

One of the maxims, which I'll reiterate here, is that "Your URLs Should Not Reveal Your Technical Capabilities". This has absolutely nothing to do with the oft-maligned "security through obscurity"—the belief that hiding information makes you less susceptible to risk. Instead, removing the technicalities of your URL will allow your site to grow with your needs.

Consider the default configuration of CGI: anytime you want to add some dynamic functionality to your site (with Perl, Python, etc.), you've got to stick your script in a directory and point to it with a URL like http://127.0.0.1/cgi-bin/add_user.pl (where .pl denotes a Perl script) or http://127.0.0.1/cgi-bin/del_user.py (.py denoting Python). What happens, though, when you no longer need CGI scripts, but have moved to an embedded language like PHP? Suddenly, all of your pages that point to that ugly /cgi-bin/ URL are broken, useless, and inaccurate. You find yourself refactoring pages to point to new locations.

The solution? Remove /cgi-bin/ and the file extension from the URL. By clearing this needless technical cruft, you get stronger addresses with less need for change. http://127.0.0.1/admin/add_user and http://127.0.0.1/admin/del_user contain no indication of the technology behind them, merely their purpose in the grande scheme of things.

We're already halfway through removing /cgi-bin/ from our URLs (file extensions I keep putting off, but I'll get to 'em eventually, I promise!). Our very first match for our "CGI" search was for an `ExecCGI` addition to the `Options` line of our root web directory (/Library/WebServer/Documents/). `ExecCGI` works similarly to `Includes`, which helped get our SSIs running last month. By adding it to the `Options` line of any `Directory` we, like SSIs, are telling Apache to allow CGI scripts at that location:

```
Options Indexes FollowSymLinks MultiViews ExecCGI
```

To make the comparison even more apt, there's one other thing we need to do before we can use CGIs wherever we need them: we need to add a handler that says "Apache! Listen up! Whenever we request a file with this extension, we want you to treat it as an executable program!" This handler was also needed last month—we told Apache that anytime it ran across a file with an `shtml` extension, it should process it for SSIs statements.

Thankfully, there's not much brainwork involved here, as our final relevant search match for "CGI" brings us to the below. Simply uncomment the `AddHandler` (by removing the #), restart Apache (`sudo apachectl restart` in your Terminal) and you're ready to go:

```
# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.
#
# To use CGI scripts:
#
#AddHandler cgi-script .cgi
```

Two caveats that may be running through head: "what about the `cgi` extension?" and "what about that security brouhaha you cautioned us about?" Concerning the file extension, yes, we're swapping /cgi-bin/ for `cgi`, but this will never be an issue once we remove the need for them in our URLs. As for security, there's not much I can say besides "be careful"—you'll no longer have the niceties of a single and central directory for your code execution, so double and triple check the scripts you'll be programming or using. I'll talk a little more about script security in next month's column.

LET'S SEE ONE OF THESE CGI THINGIES IN ACTION!

Browse to your /Library/WebServer/CGI-Executables/ directory, the default location that Apache has been configured for CGI scripts. You should see two files, `printenv` and `test-cgi`, there already. These are some default Apache CGI scripts that will help you quickly test if things are a-ok. Since CGI was already pre-configured in the `httpd.conf`, we should be able to just load one of these files in the browser and see some fireworks, right? Load <http://127.0.0.1/cgi-bin/test-cgi>, and we'll receive the response in **Figure 1**.

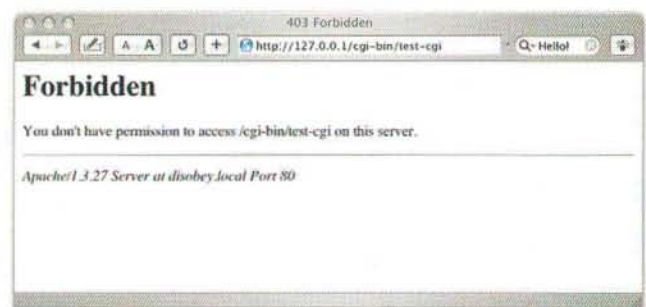
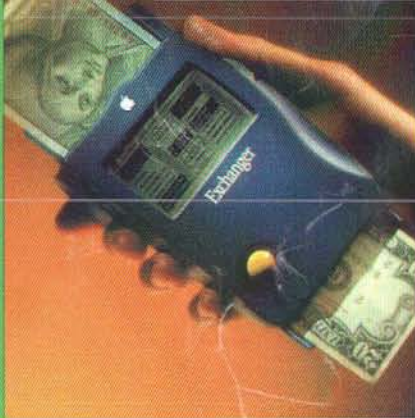


Figure 1: An error after loading our `test-cgi`, but why?

32
< 1 year
2 years >
\$64



INTERVIEWS

TAPPING INTO THE WORLD OF CELEBRITIES AND THEIR MACS, ONLY MACDIRECTORY OFFERS EXCLUSIVE INTERVIEWS. GET A CLOSE AND PERSONAL VIEW FROM SARAH JESSICA PARKER, STING, STEVE JOBS, MADONNA, HARRY CONNICK JR., GEORGE LUCAS, JENNIFER JASON LEIGH, STEVE WOZ AND OTHER LEADERS IN THE MAC COMMUNITY.



FEATURES

DESIGNERS, WRITERS, MUSICIANS, BUSINESS LEADERS & OUR TECHNICAL EXPERT TEAM OFFER THEIR OWN PERSONAL INTERPRETATION OF THINGS THAT ONLY THE MAC SYSTEM CAN DELIVER. FEATURING OVER 240 PAGES OF REVIEWS, INTERVIEWS, NEWS, INSIGHTS, TRENDS AND THE LARGEST MACINTOSH BUYERS' GUIDE INCLUDING OVER 5,000 MAC PRODUCTS AND SERVICES.



CULTURE

MACDIRECTORY TAKES YOU TO THE WILDEST CORNERS OF THE WORLD AND UNCOVERS HOW MACINTOSH COMPUTERS ARE BEING USED BY OTHER CULTURES. TRAVEL TO JAPAN, AUSTRALIA, GERMANY, BRAZIL, INDIA, RUSSIA & LEARN MORE ABOUT APPLE'S CULTURAL IMPACT AROUND THE GLOBE.

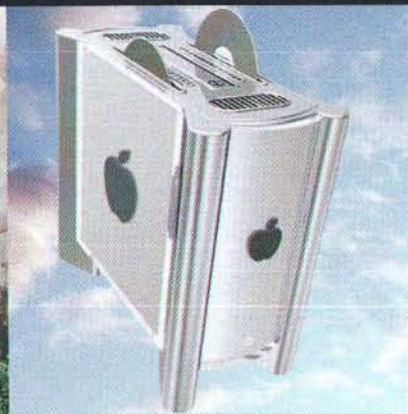
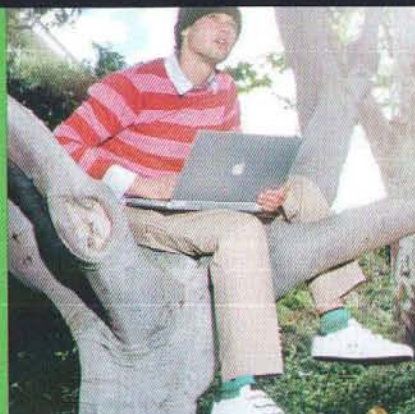
MacDirectory

BEYOND ANY MACINTOSH MAGAZINE. SUBSCRIBE.

< Subscribe

www.macdirectory.com/mw.html

SEND CHECK OR MONEY ORDER TO:
MACDIRECTORY SUB DEPT.
326 A STREET, 2C
SOUTH BOSTON, MA 02110



REVIEWS

FIND OUT ALL YOU NEED TO KNOW ABOUT THE LATEST MAC PRODUCTS INCLUDING THE HOTTEST MAC OS SOFTWARE AND HARDWARE.



WIN!

SUBSCRIBE TO MACDIRECTORY AND YOU WILL AUTOMATICALLY ENTER OUR SWEEPSTAKES FOR A CHANCE TO WIN A NEW TITANIUM!

If CGI scripts were already configured, why this rather rude error message? Let's check Apache's `error_log`, which will always contain some sort of light bulb enlightening response. After running `tail /var/log/httpd/error_log` on the command line, we'll see something similar to the below as the last line of output:

```
[Sun Aug 17 10:41:16 2003] [error] [client 127.0.0.1] file
permissions deny server execution: /Library/WebServer/CGI-
Executables/test-cgi
```

This is one of the more common errors you'll experience with CGI scripts: the `test-cgi` file doesn't have the proper credentials to be considered an executable program. I won't get into the vagaries of file ownership or permissions, but follow through the bolded commands below to give both of the default CGI files "execute" access for all users:

```
> cd /Library/WebServer/CGI-Executables/
> ls -l
total 24
-rw-r--r--  1 root  admin   5398 Jul 27  2002 printenv
-rw-r--r--  1 root  admin    757 Jul 27  2002 test-cgi
> sudo chmod 755 *
Password: *****
> ls -l
total 24
-rwxr-xr-x  1 root  admin   5398 Jul 27  2002 printenv
-rwxr-xr-x  1 root  admin    757 Jul 27  2002 test-cgi
```

Now, let's try our URL again. **Figure 2** shows our new output:

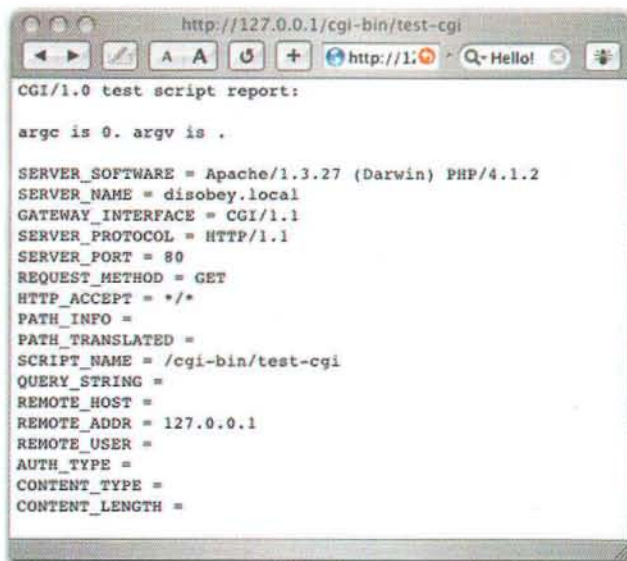


Figure 2: The correct output of our `test-cgi` script.

Figure 3 shows the output from the other script, `http://127.0.0.1/cgi-bin/printenv`:

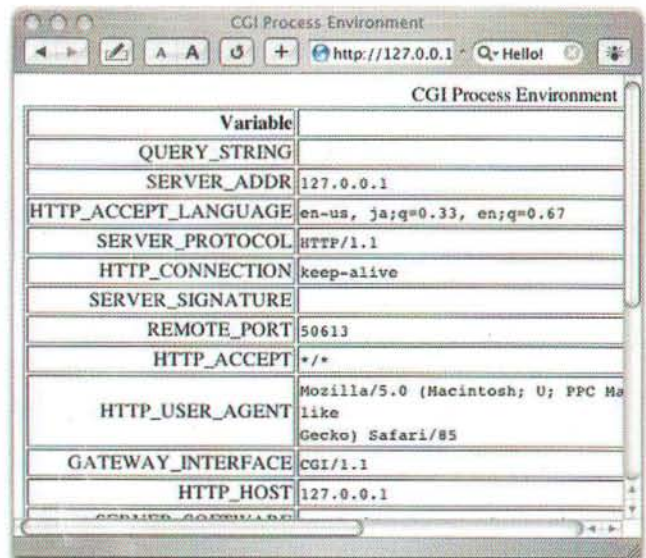


Figure 3: Similar output from the default `printenv` script.

If you followed the instructions under the "CGI EVERYWHERE!" subheading, you can also move either of the `printenv` or `test-cgi` scripts out of their current location and stick them in `/Library/WebServer/Documents/` with a `cgi` file extension. Once you do so, you should then be able to run them under the non-`/cgi-bin/` URL and receive the same output (if not, what's your first course of action? Check the `error_log`!).

Assuming you get similar output as **Figures 2** and **3**, what does all this gibberish mean? What exactly did we just do? Both scripts show a number of *environment variables*, which is just a fancy term for data that exists "invisibly" and floats around waiting to be used or read. Some of the variables were set by the browser (like `HTTP_USER_AGENT`), and some have been set by the server (like `SERVER_PROTOCOL`). When I first introduced CGI, I mentioned that Apache "sends some relevant information to the requested resource". This "relevant information" is stored in environment variables.

Environment variables can exist outside of your web server too. For instance, if you run `printenv` in your Terminal, you'll see a listing of variables similar to **Figure 4**:



Get MacTech delivered to your door at a price **FAR BELOW** the newsstand price. And, it's **RISK FREE!**

Subscribe Today!

www.mactech.com

```

Terminal — bash — bash (tty1) — 90x24
morbus@disobey:~$ printenv
PWD=/Users/morbus
CVS_ROOT=d:text:morbus@cvs.amphetadesk.sf.net:/cvsroot/amphetadesk
SQLL_CATALOG_FILES=/sw/etc/sql/catalog
XML_CATALOG_FILES=/sw/etc/xml/catalog
GDK_USE_XFT=1
PERLSITE=/sw/lib/perl5
TERM_PROGRAM=Apple_Terminal
MANPATH=/sw/share/man:/usr/share/man:/usr/local/man:/usr/X11R6/man
USER=morbus
CVS_RSH=ssh
TERMCAP=
INFOPATH=/sw/share/info:/sw/info:/usr/share/info
SHLVL=1
SHELL=/bin/tcsh
HOME=/Users/morbus
TERM=vt100
TERM_PROGRAM_VERSION=61
PATH=/sw/bin:/sw/sbin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/db
l/ite/bin:/usr/local/sbin:/Users/morbus/Applications:/Users/morbus/Conquest/Disobey.com/de
tergent/code:/Users/morbus/Conquest/Disobey.com/detergent/code/leecharoo:/usr/X11R6/bin
__CF_USER_TEXT_ENCODING=0x1F5:0:0
~/usr/bin/printenv
morbus@disobey:~$

```

Figure 4: Environment variables in your Terminal.

Believe it or not, we've dealt with environment variables already in this series. In our article on Server Side Includes, we checked the \$QUERY_STRING variable to see which quote should be displayed at the user's request. If you check back to Figure 3, you'll see an environment variable named QUERY_STRING, which is currently empty. Try adding ?q02 to the end of our URL (<http://127.0.0.1/cgi-bin/printenv?q02>), and see how that changes the output (Figure 5):

Variable	Value
QUERY_STRING	q02
SERVER_ADDR	127.0.0.1
HTTP_ACCEPT_LANGUAGE	en-us; ja;q=0.33; en;q=0.67

Figure 5: Our QUERY_STRING now has a value.

With the above two scripts working correctly, our CGI capabilities are ready for use.

HOMEWORK MALIGNMENTS

In our next column, we'll chat further about CGI: how to begin creating our own, the most common errors during development and deployment, security issues we should be aware of, and how to find the "good ones" from the immense ocean of crap. If we have time, we'll show you how to install a few that have stood the test of time. For now, students may contact the teacher at morbus@disobey.com.

- Take a look inside printenv and test-cgi in a text editor.
- View the printenv script from various browsers and machines. Determine how the output changes, and see if you can figure out what each setting actually means.
- Forgive me for not including anything amusing in this article.

"Well, what're you waitin' for? Draw!"

Eazy Draw

Make Drawing Fun on OS X



Introducing EazyDraw – the fun, easy-to-use Mac OS X design tool that lets you draw like a pro! Now you don't need to be a graphic artist to create great illustrations. EazyDraw's vector-based graphics and editing capabilities make it easy to create technical diagrams, flow charts, and business communications as well as commercial line art illustrations and graphic elements for application software and web design.

Learn more about EazyDraw today! Get big savings buying direct from our online store. Visit us at **www.eazydraw.com** (That's easy with a Z).

© 2003 Dekorra Optics, LLC. All rights reserved. EazyDraw and the "box of tools" are trademarks of Dekorra Optics, LLC. Mac and Built for OS X are trademarks of Apple Computer, Inc.



By Dave Kelly

Automate your testing... with Eggplant

Exploring a new tool for performing automatic tests

TEST AUTOMATION ON THE MAC – A BRIEF OVERVIEW

Your code has bugs in it. What? So, you don't like hearing that, eh? But it's not your fault. All software has bugs. Finding all the bugs in your software is a big job so you need all the help you can get, right?

Bug finding is an art that complements code writing. Somewhere along the way someone (probably an overworked quality engineer) figured out that time and effort could be saved by using automated testing. There are several automation tools available for testing Windows software including software from Rational, Mercury Interactive, Segue Software, and others. But, until recently there was nothing that come close for Mac OS X. In Mac OS X, the tester has had to rely only on AppleScript or Perl for test automation. It doesn't take too long to realize that although they are very powerful tools, neither AppleScript nor Perl are adequate for testing some things. GUIs and functionality that can only be accessed through the software being tested to name a few. Even the new GUI Scripting (System Events) beta software (see <http://www.apple.com/applescript/GUI/>) does not allow the GUI of some software to be tested. We're now going to take a look at Redstone Software's Eggplant software for testing Mac OS X software.

Redstone Software, Inc. is a subsidiary of Gresham Computing plc, which is based in Southampton, UK. Gresham Computing (GHT) is traded on the London Stock Exchange. In late 2001, Gresham Computing started work on a new product internally named Operation Screaming Eggplant. Their goal was to deliver a product with high ease-of-use for the neophyte yet still offer powerful capabilities for the experienced tester. The Mac OS X platform was chosen as the base OS for this product for two major reasons. First, they wanted to establish the product before competing in the PC marketplace where competition is much more fierce. Second, they believed that Mac OS X had the right features and extensibility to support the enterprise market.

Redstone Software, Inc. was created in early 2002 and in October of 2002 became the first company to deliver a cross-platform automation tool to the Mac OS X platform. They stuck

with the code name and named their product Eggplant. Their belief is that automation products of this kind are traditionally too complex and require specialized development skills that are rarely found in testers. In addition, they also believe that very few automation projects have been successful, although there are cases where automation has worked. In these cases automation is performed for tests requiring special or repetitive computations, such as acceptance tests, where test execution and results are predictable.

It might be interesting to note that Redstone Software uses eXtreme Programming (XP). This is very apparent by their willingness to listen to and support their customers. Through this intense support of their customer's needs, they released several versions of Eggplant—the most recent on being version 1.3 in May 2003. The remainder of this article will focus on Eggplant v1.3.

The Eggplant software runs on your host Macintosh. This is the computer that controls the test. The software you are testing runs on a separate computer called the "System-Under-Test" (SUT). The SUT runs Virtual Network Computer (VNC) server software. VNC is a remote display system that allows you to view the desktop of the "server" computer on another computer anywhere on the Internet. The VNC server software could be running on another Macintosh or it could be running on a Unix or Windows computer. Eggplant is able to connect to the SUT using the VNC server software.

VNC originated from AT&T Laboratories Cambridge's development of very-thin-client ATM network computers called the Videotile (see <http://www.uk.research.att.com/tile.html>). The VNC viewer and server are basically software-only versions of this ATM Network Computer. VNC is free and is distributed under the terms of the GNU Public License. Eggplant incorporates VNC client software to give it full access to the SUT GUI, keyboard, and mouse using any VNC server. Any VNC client "viewer" can display the desktop of any VNC server; however, to use Eggplant you'll have to use the client that is part of Eggplant itself. VNC clients are available for every major OS including: Mac OS 9, OS X, windows 95/98/ME/2000/XP, UNIX (AIX, Solaris, HPUX), and LINUX (see <http://www.uk.research.att.com/vnc/>). The Redstone VNC server, OSXvnc 1.2 (see <http://www.redstonesoftware.com/osxvnc/>) is recommended for use with Eggplant because of fixes made that improve performance and increase stability.

Dave Kelly is currently a Software Quality Engineer at EarthLink and has many years of software testing experience at EarthLink, Hewlett Packard, Symantec and Xerox. He has worked and played with the Macintosh since 1984 and was one of the founding editorial board members of *MacTutor Magazine* (now *MacTech Magazine*). You can reach him at dkelly@earthlink.net.

The setup for running Eggplant consists of just starting up the 496KB OSXvnc application and starting the server by clicking the "Start Server" button (see **figure 1**). The OSXvnc application will need to remain running or the connection will be broken. You'll also want to hide the OSXvnc application before running your test.

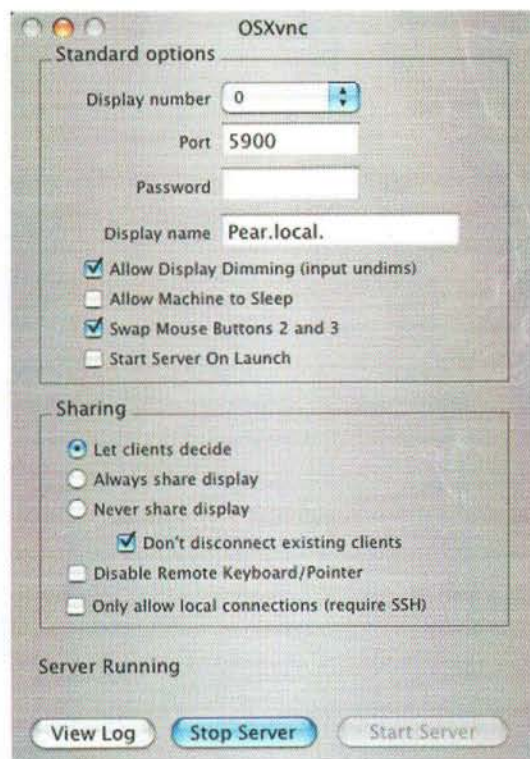


Figure 1. OSXvnc Interface

Although the real testing happens on the SUT, the host Macintosh is where the action is. When you open Eggplant the first thing you'll want to do is connect to the SUT, if anything else just so you can see how it works. After entering your license you'll see a window that prompts you to establish a connection with a VNC server. You can connect with the IP address, port and password of the SUT. It is also possible to connect using IP over Firewire or through an SSH connection.

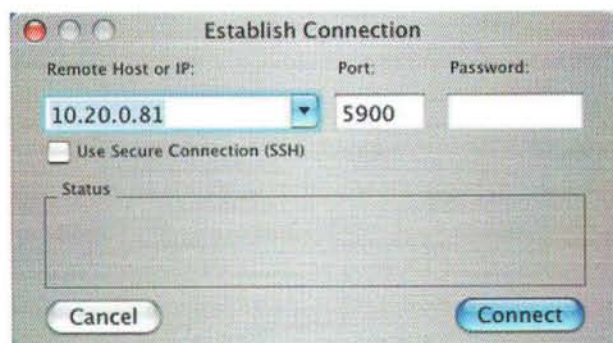


Figure 2. Eggplant connection window

Long Distance

3.9¢ Per Minute!

Straight 6 second billing increments

Excellent rates on intrastate, intralata/toll calls and international calling with no term contract.

Toll Free (800/888/877/866) service, same low per minute rate for incoming calls.

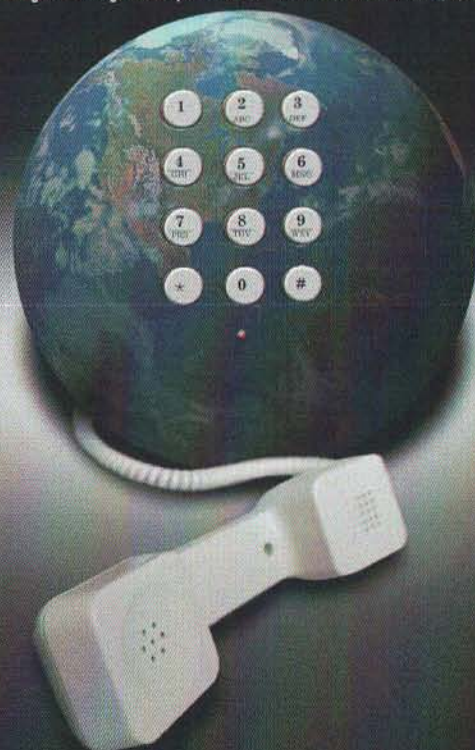
10 cents per minute calling card.

Detailed billing directly from Capsule Communications, a Covista Company.

Quality electronic and telephone customer support.

No monthly billing fee if you sign up for AUTOPAY billing option or if your bill is over \$20.00 each month.

(NOTE: \$1.00 billing fee is charged when your bill is under \$20.00 for all non-Autopay customers.)



www.lowcostdialing.com

Once the connection is established, you see the Eggplant monitor and can control the SUT. The live remote screen in **Figure 3** shows the desktop of the SUT.

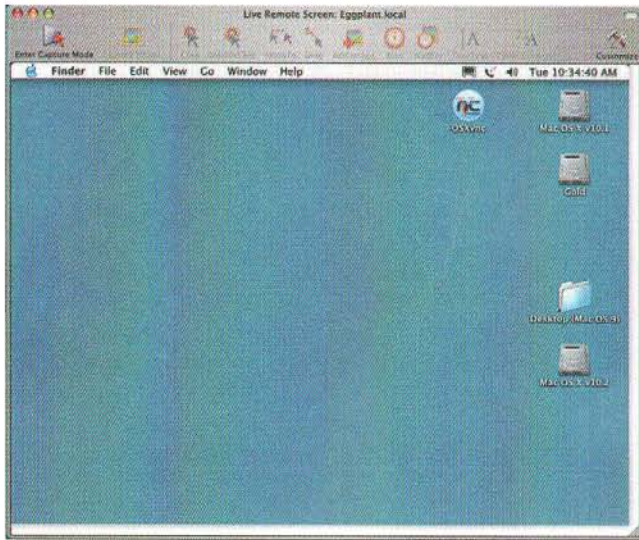


Figure 3 Eggplant Remote Screen viewer

At this point, the VNC viewer can only control the remote SUT and capture images. That might be useful for some things but you'll need to create a script in order to begin writing a test. You start by creating a suite. When a new suite is created, Eggplant creates a directory structure to store the scripts, images, and log files pertaining to that suite. The suite window in **figure 4** is used to access all of the features of the suite.

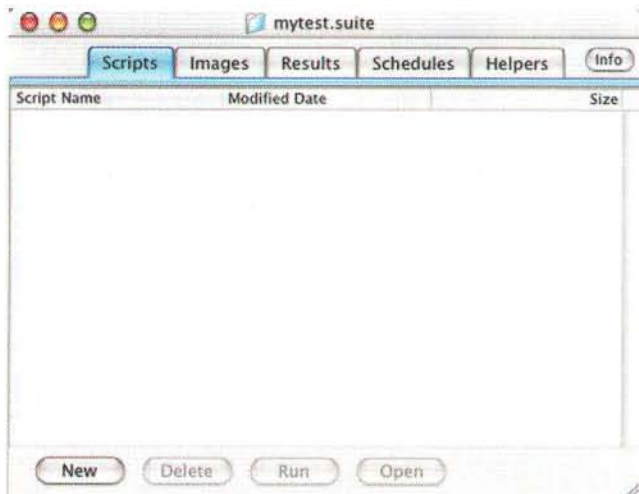


Figure 4 Eggplant suite window

Now for the heart and soul of the Eggplant... capturing an image. Once you open a new script you're ready to start capturing. Building test scripts is dependent on being able to capture images through the viewer window. Without capturing

images, there isn't much that you can do.

The viewer has two modes, Capture Mode and Live Mode. Eggplant starts out in live mode where you can see and control whatever goes on with the SUT. By selecting "Enter Capture Mode", you are able to select a rectangular section of the SUT's screen through the Eggplant viewer, along with a hot spot, represented by a red crosshair (see **figure 5**). The hot spot represents the point that will be clicked when the script is run. The rectangular section is the area that will be captured as a snapshot. After selecting the image and setting the crosshair to the point you want to click, you can choose between standard commands that will move the mouse and click or type text on the SUT's screen. The standard commands that can be used are Click, DoubleClick, MoveTo, Drag, Add Image, Wait, WaitFor, TypeText and TypeCommand. These pretty much cover anything you will need to do to test your software.



Figure 5 icon selected and ready to capture

After selecting your image, setting the click point, and selecting a command (example "DoubleClick"), Eggplant will add the image to the suite's set of images and add the command to the script. For this example, this line would be added to the current script window:

```
DoubleClick "image0001"
```

Meanwhile, the image is added to the suite's images as in **figure 6**.

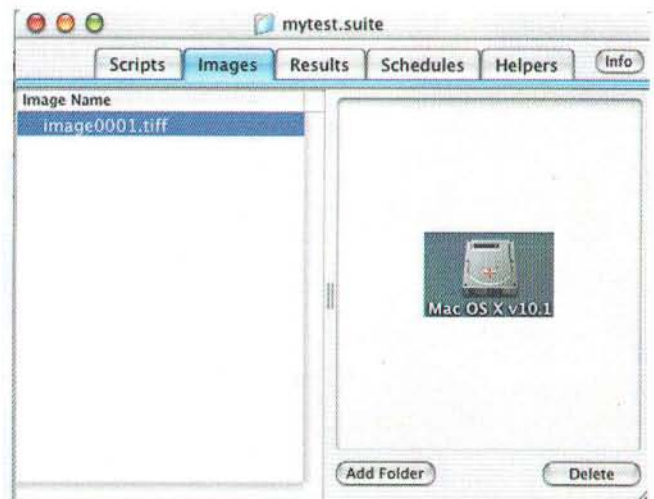


Figure 6 suite images

After capturing a sequence of images/commands, you can go back and establish your initial setup conditions and run the script (or select a portion of the script to run). The viewer screen is compared to the saved image for each command and if it finds a match, the command is executed. If it fails, the failure is stored in the suite's Results log. A log is stored for each time you run the script. During execution you can also log your own results from the script.

It should be apparent that the key is having the saved image match the captured image when the script runs. Eggplant does a great job of finding the correct image on the SUT's screen, but there are certain circumstances you must be aware of. For example, if the desktop background looks like a solid background, but isn't really a solid background, then the image won't be found if it is not exactly where it was when the saved image was captured. The solution then is to make sure that the captured images are over a real "solid" background. Also, it is helpful to only capture the minimum that is required to identify the image being compared. For example, it may only be necessary to capture the text of an image, or maybe just the center portion of the image. It only needs to be enough to be identified as a unique image.

Eggplant is very forgiving when matching captured images. When images are captured, Eggplant attempts to automatically detect the right setting for color matching. In most cases it defaults to "Tolerant" which means that it will allow small variations in the color to be accepted. The "Precise" setting will look for a more exact match in the colors. Then there is the special "Pulsing" setting that allows color matching even when the color is a pulsing Mac OS X Aqua button. It's a good thing they included these color matching options so images, such as pulsing buttons, could be matched.

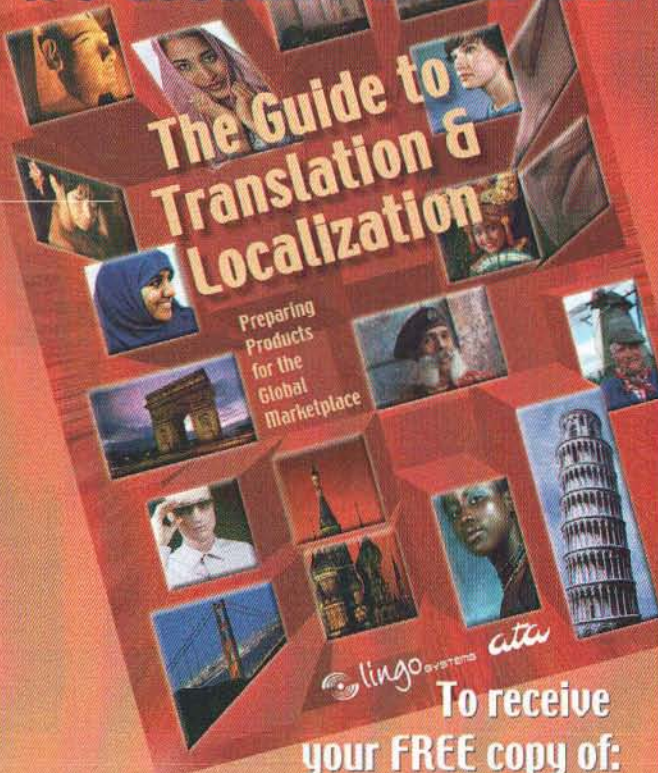
IN SEARCH OF THE "RIGHT" AUTOMATIC TESTING TOOL

It will help if you know what you're looking for in an automatic testing tool. Regrettably, there isn't much available for Mac OS X to compare with when looking for automation tools. It would still be worthwhile to know what you want the test environment to provide and what you're going to have to add to it to make it work for you. The following is a short list of "stuff" to look for in test automation software and comments about how well Eggplant does in each of the areas listed.

- Will the test tool interfere with the software under test? The VNC server on the SUT is not very intrusive so you'll hardly even know it's there. You'll have to know your system under test well enough to know if a VNC server would have a material impact on it.

Eggplant relies on an IP connection. If your software makes changes to network settings Eggplant's connection with the SUT could be broken. This behavior needs to be taken into account when writing scripts.

Classic or Cocoa Applications? We Localize Them All!



To receive
your FREE copy of:
**The Guide to Translation and
Localization – Preparing Products
for the Global Marketplace**

Call: 1-800-878-8523 Email: info@lingosys.com
Fax: 1-503-419-4873 Or visit: www.lingosys.com

**We focus on what matters most: meeting your
needs and providing excellent customer service.**

- Translation
- Desktop Publishing
- Project Management
- Localization
- Engineering
- Quality Assurance



15115 SW Sequoia Pkwy. #200 Portland, OR 97224

- Can the test be controlled by scripts? Eggplant has a built in test script language called SenseTalk. SenseTalk is a HyperTalk-like language. The language is quite powerful so you should be able to do many of the things you need to do. The script is run on the host Macintosh (not the SUT) so all of the local commands refer to the host. For example, you can read/write to a local disk file (you might do this to record some test results or get data for a test from a file), but the file must be available to the host computer. If you need access to the SUT's files, there is an approach that may be acceptable. The workaround is to turn on file sharing on the SUT and mount the SUT's disk on the host computer.
- Can scripts be captured or do I have to modify scripts after capturing them to make them work? Since images are an integral part of the commands it is necessary to capture the images that will be compared. You cannot just start typing without also capturing an image. Some automation tools will claim that all you have to do is capture your script and then run it and you're all set. Eggplant is really very different because you are not capturing your script as much as you are capturing images that the script will use. Even with Eggplant scripts don't write themselves (it is wishful thinking when any company claims their tools will write the scripts for you). With Eggplant, the process of writing the scripts includes creating the images that will be used to compare against when the script runs. Fortunately, this process makes it very easy to get the script up and running.
Redstone Software makes the claims "Manual test-oriented commands allow the non-programmer to quickly create scripts. Eggplant also offers the flexible and robust SenseTalk scripting language and the ability to test multiple applications and platforms", "Eggplant interactive script development simplifies test automation" and SenseTalk scripting language provides high ease of use and powerful capabilities". Unlike other automatic software test applications, they make no claims that the script can be captured and run without modification. There will be a certain amount of script writing to get your tests up and running, but as they claim, Eggplant simplifies script development.
- Can scripts be reused? Test scripts can call other test scripts. In this sense, "libraries" of scripts can be written to be used over and over. In addition, scripts belonging to other suites can be called by specifying them in the Helper Suites list.
- Can scripts be scheduled to run at a certain time? The scheduler can setup a selected batch of scripts so that they will run on specific SUTs, but you are not able to set up a specific time for a script to run. You could set up a script that checked the time before it did anything though. Redstone recommends use of the cron function to schedule tests. The batch of scripts can also be "scheduled" to repeat however many times you specify.
- Can scripts be easily printed/documented? Yes, scripts can be printed. However, images cannot be printed from the Eggplant application. The images are stored with the suite as .tiff files, and those can easily be printed if necessary.

Additionally, scripts can contain comments to further document what is being done.

- Is there a test infrastructure in place? Not completely. The test suite environment of Eggplant includes a script editor, image list (showing images included in the suite), results list, script scheduler, and Helper Suites list. Eggplant could be improved to include a better debugging environment, the ability to link test requirements to tests being run, and an expanded capability to keep track of test results.
- Is there a mechanism in place to keep track of test status? The Results tab is actually a very nice system for keeping track of test status. Building a test logging mechanism can be a very time consuming task. Fortunately, with Eggplant you don't need to build anything, and the logging is built right into the script language. Unfortunately, there isn't a way to print the results from within the Eggplant application. But, fortunately, the file is stored as a plain text log file that can be easily printed.
- Is there an extra cost per user? Can multiple tests be run concurrently? If you want to run multiple test streams against different SUTs at the same time, then that requires multiple licenses. The current pricing scheme for multiple licenses on a single host is as follows. The base license allows execution of one Eggplant test stream for \$3400. A second test stream costs \$1,700 and the third test stream adds \$850. For \$5,950, one system can execute three different Eggplant test streams. Four or more concurrent stream licenses are \$425 each. A lower end Mac can easily handle three streams; higher end Macs can of course run more. What many users do is to use one Mac as an Eggplant execution system and have licenses on other machines for script development.
The Eggplant pricing is based on the number of test streams being run with no additional charge for the number, location, or type of systems-under-test.
- Is there any debugging capability built-in? Although there is no ability to add breakpoints, any portion of a script can be run at any time by selecting it and clicking on "Run Selection" in the script editor window.

SUMMARY

Eggplant is a fine application. The really cool thing about it is that it can run tests on multiple platforms (all you need is the VNC server for the SUT you're testing on). Please don't think that Eggplant is for software testing only. Developers can automate the build process. System Administrators using Eggplant can automate server maintenance. Any repetitive task is a candidate for automation. This is the first really usable automatic software tool that has been available for Mac OS X. It is very depressing to hear other companies tell you "Sorry, we don't support the Mac". Well, this changes everything because now there's a tool that works with Mac OS X.

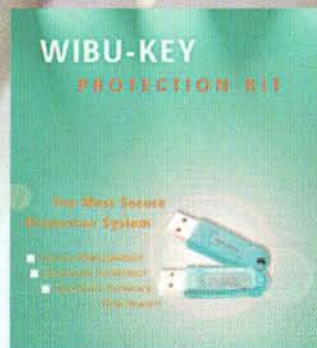
BIBLIOGRAPHY AND REFERENCES

- Kelly, Dave. "Software Automation and the Product Life Cycle". *MacTech Magazine* (formerly *MacTutor*) 13:10.
- Cem Kaner, Jack Falk, Hung Quoc Nguyen. "Testing Computer Software". 2nd edn. New York 1999.

WIBU-KEY Software Protection

Be sure to order a **WIBU-KEY Protection Kit** and find out why small software companies to Fortune 500 enterprises are switching to the WIBU-KEY Security, not Obscurity model for all of their software licensing needs, including:

- Common API across all platforms and hardware
- Hardware-based code and data encryption
- Field-upgradable and reusable hardware
- The most cost-effective network licensing solution available
- The only system to employ RID/RED and AXAN security
- Engineered and manufactured to exacting ISO9001 standards



Test the
WIBU-KEY
Protection Kit
sales@griftech.com

Call 800-986-6578

The Key is in Your Hands!

WIBU
SYSTEMS

WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
Email: info@wibu.com

www.griftech.com
www.wibu.com

Protection Kits also available at:

Belgium wibu@impakt.be, Denmark leam@darbit.dk, Finland finbyte@finebyte.com, France info@neol.fr, Hungary info@mrsoft.hu, Japan info@suncarla.co.jp, Jordan starsoft@cyberia.net.lb, Korea dhkim@wibu.co.kr, Luxembourg wibu@impakt.be, Netherlands wibu@impakt.be, Portugal dubite@dubit.pt, Thailand preecha@dpf.co.th, United Kingdom info@codework.com, USA sales@griftech.com

by Tim Monroe

Revolution

Developing QuickTime Applications with Revolution

INTRODUCTION

Revolution is a software development environment written and published by Runtime Revolution Ltd., a company based in Edinburgh, Scotland. Revolution is a rapid application development (RAD) tool that can build applications for a number of operating systems, including Mac OS X, classic Macintosh OS, Windows, and several versions of Linux and UNIX. In this article and the next, I want to take a look at using Revolution to build applications that can open and display QuickTime movies. Accordingly, we'll limit our attention to using Revolution on Mac OS X and Windows systems.

As usual in our recent articles, our goal is to build a multi-window movie playback and editing application. Let's call this application RunRevVeez. Ideally, the movie windows displayed by RunRevVeez should be indistinguishable in appearance and behavior from movie windows displayed by our other sample applications (such as the C-based application QTShell or our REALbasic application BasicMovie). And, ideally, RunRevVeez should be able to invoke any of the thousands upon thousands of QuickTime APIs.

I'm happy to report that Revolution performs quite nicely in both regards. It's extremely easy to use Revolution to open and display QuickTime movies, and to get them to behave as expected. Revolution provides a reasonable selection of built-in commands and functions for working with movies and movie properties. And Revolution provides a mechanism — using plug-in code modules — to access features of QuickTime that are not already built-in to Revolution.

In this article, we'll see how to set up a new Revolution project for a QuickTime player application. We'll add an About box to the application, and we'll see how to create new movie windows and load movies into them. We'll create the application's menus and add executable code to a few of those menus. By the time we reach the end of this article, we

should have a pretty good understanding of what it's like to create a simple application with Revolution and of how to work with QuickTime movies in that application. And everything we do in this article should work as well on Windows as it does on Macintosh.

In the next article, we'll concentrate on implementing those features of RunRevVeez that are not at all (or not very easily) supported directly by Revolution. For instance, Revolution does not currently provide any methods or commands for editing a movie (cutting or copying the movie selection, pasting a previous cut or copy, and so forth); but it's easy enough to write a plug-in module that supports these capabilities, and it's easy to access that module from within Revolution code. In fact, writing and calling Revolution plug-ins is so easy that I can't imagine any serious programmer not wanting to use them as a matter of course. So it will be good to take a careful look at this technique. In that article, we'll focus primarily on development on Mac OS X.

The current released version of Revolution is version 2.0.2, and that's what I've used throughout these articles. This version incorporates a large number of improvements over previous versions.

REVOLUTION OVERVIEW

Revolution, like any good RAD tool, unifies the development of an application's user interface and its executable code within a single environment. User interface elements (such as buttons, scroll bars, text-edit fields, pop-up menus, and so forth) are objects that can be placed into windows and dialog boxes; these objects have properties that we can get and (usually) also set. We do this, and other operations, by attaching code to objects that is executed when particular events occur to those or other objects. Revolution is not event driven, however, at least in the classic Mac sense of event-driven programming that involves a global event loop. Rather, Revolution is *message driven*: all executable code is contained in message handlers that are attached to objects in the application. There are, for instance, start-up and open messages that are sent when a window, dialog box, or even the application itself is opened.

Tim Monroe is a member of the QuickTime engineering team at Apple. You can contact him at monroe@mactech.com. The views expressed here are not necessarily shared by his employer.

So far, this is part of any standard RAD diet, and it is not terribly unlike the programming model we saw in AppleScript Studio or REALbasic (covered in earlier *QuickTime Toolkit* articles). Revolution is interesting in several ways. First, it uses a so-called fourth-generation language for its message handlers. This language is called *Transcript* and is a clear descendent of the HyperCard family of languages. In Transcript, there are no explicit type declarations; the values of all variables are treated as strings but are coerced to other types when necessary. For instance, we can add two values together like this:

```
put 8 + 11 into mySum
```

After this code is executed, the variable `mySum` contains the value "19" (a string). To concatenate two strings, we use the "&" operator, like this:

```
put 8 & 11 into mySum
```

After this line of code is executed, the variable `mySum` contains the value "811".

Revolution is interesting and distinctive also in the basic vocabulary it uses to describe the application and its

interactions with a user. (Again, this is a holdover from HyperCard.) A window is a *stack*. Every application has one *mainstack*, and all other stacks in the application (that is, all other windows and dialog boxes) are substacks of the mainstack. Typically the mainstack serves as the application's main window. For document-based applications (like our sample application RunRevVeez), there might however be no such main window; in this case, it is still useful to have a mainstack, since messages targeted at a substack are redirected to the mainstack if the substack does not contain an appropriate message handler. In this case, the mainstack can just be invisible.

Stacks contain *cards*. A card occupies the entire content area of the stack that contains it. Cards, in turn, contain *controls* — which are the basic user interface objects like buttons, scrollbars, etc. A stack can contain more than one card, and a card can contain more than one control. (Only one card in a stack can be visible at a time.) Controls can be grouped, for easy reference. For our purposes, we'll need to work with only one group of controls, the menu bar menus.

THE PROJECT

Let's get started building our movie playback and editing application, RunRevVeez. First, launch Revolution. After the



effigent, Inc.



We deliver efficient and intelligent business solutions

Providing Server and Client enterprise applications for the Mac platform

Services

- Application Development
- Conversions to Mac
- Testing Services
- Maintenance & Support

Solutions

- E - Commerce
- Education Portals
- Knowledge Management
- ERP Implementations

Technology

- Cocoa
- Mac OS X
- WebObjects & J2EE
- Quicktime & Streaming



info@effigent.com

www.effigent.com

Contact Us: 760-723-4800

Copyright 2003, effigent, Inc. All rights reserved.

splash-screen disappears, two windows appear, a *tools palette* (**Figure 1**) and an *application browser* window (**Figure 2**). The application browser currently shows no stacks in the application.



Figure 1: The tools palette

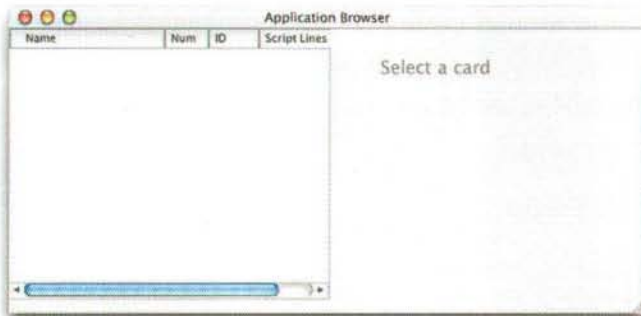


Figure 2: The application browser

The tools palette shows icons representing a number of controls that can be placed into cards, including push buttons, tabbed buttons, check boxes, radio buttons, text-edit fields, list boxes, scrollbars, pull-down and pop-up menus, and images. Of particular interest for us in this article is the *player object* in the lower-right corner (whose tool tip is “QuickTime Player”).

The top two icons in the tools palette do not represent controls. Instead, they indicate which mode the IDE is in.

When the arrow is active (as in **Figure 1**), the IDE is in *edit mode*: we can add stacks, place controls into cards, and edit the executable code in a message handler. When the little hand icon is active, the IDE is in *browse mode*: we can run scripts and interact with the objects in a card. By entering browse mode, we can test out the operation of our stacks and scripts without having to build the application or leave the IDE.

Adding Stacks

The first thing we need to do is add a new mainstack to the application. Select “New Mainstack” in the File menu. A new empty stack opens, as shown in **Figure 3**.



Figure 3: The new, empty mainstack

In earlier articles, we used the default new window provided by the programming environment as a template for the multiple movie windows our applications could open, and indeed it's tempting to just click the player object icon in the tools palette and then click-drag a new movie player in the new window. With Revolution, however, we're going to use a slightly different architecture in our application. It will be easiest to add another stack to the application and to use that substack as the template for our movie windows. The mainstack, for the moment, will serve merely to hold global handlers (as mentioned briefly above).

Let's change the name of the mainstack. Select the “Stack Inspector” item in the IDE's Object menu to open a *properties inspector* like the one shown in **Figure 4**.

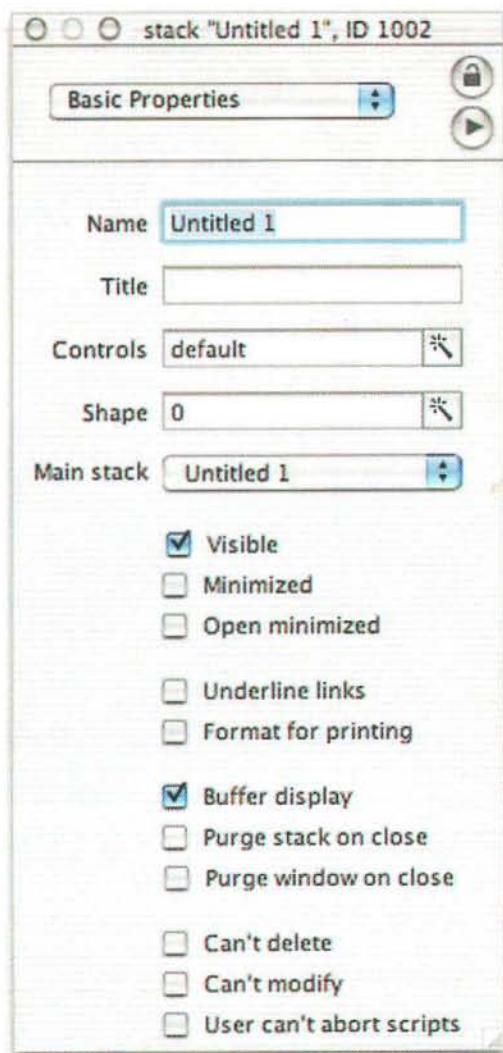


Figure 4: A properties inspector

Set the Name field to "RunRevVeez"; notice that the name of the mainstack in the pop-up menu also changes to "RunRevVeez".

Next let's take a look at how we can attach a message handler to the mainstack. In the properties inspector, press the button with the triangle icon near the top of the window; this pops up a *selection menu*. (See Figure 5.) Select the "Edit Script" item in that menu.

MacTech
M A G A Z I N E

Get MacTech delivered to your door at a price **FAR BELOW**
the newsstand price. And, it's **RISK FREE!**

Subscribe Today!
www.mactech.com



IP*Works!
The Only Truly Comprehensive
Internet Toolkit

- More Than 30 Components Cover All Major Internet Protocols
- Follows Exact RFC Specifications
- Market-Tested For Over 7 Years
- In Use By Almost All Fortune 500s
- Royalty-Free Pricing

*IP*Works! for Mac OS X (10.0/10.1/10.2) gives you access to a host of new capabilities that will connect your applications to the Internet with unprecedented ease of use, power, and flexibility! You will be able to easily and quickly:*

- program the web: HTTP, WebForm, WebUpload
- call web services: SOAP, XMLp
- transfer files: FTP, TFTP
- send email: SMTP, FileMailer, HTMLMailer
- receive email: POP, IMAP
- read/post news: NNTP
- encode/decode: MIME, NetCode
- access directories: LDAP
- manage networks: SNMP, Whois, Ping, TraceRoute
- build clients and servers: IPPort, IPDaemon
- build packet applications: UDPPort, Multicast
- remote access: Telnet, Rexec, Rshell, RCP

...and a whole lot more! - download your free trial today from www.nsoftware.com!

IP*Works! for MacOS X is currently available as a C/C++ Library (OS X Framework). Objective-C Classes for Cocoa and RealBasic plugins coming soon at www.nsoftware.com

STAY TUNED!

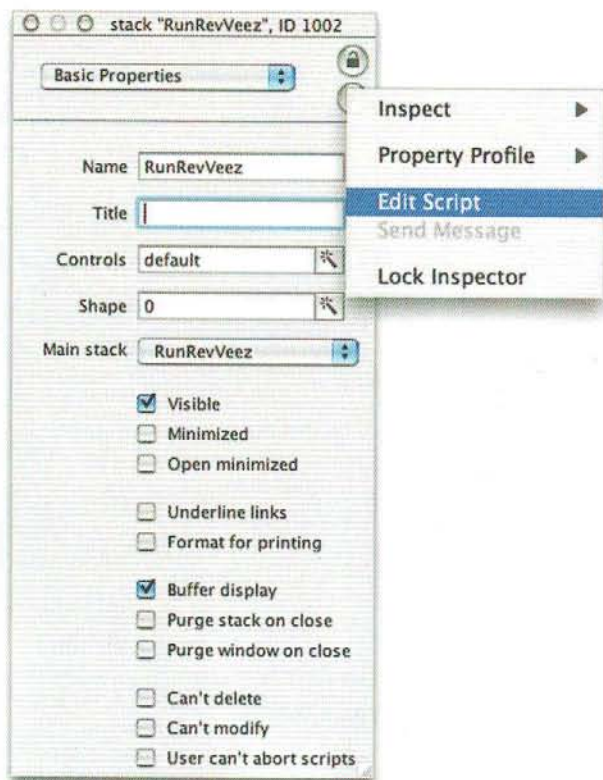


Figure 5: The selection menu

At this point, a *script editor window* will appear, as shown in Figure 6.



Figure 6: The script editor window

We can add message handlers in this window. For now, let's type in the code shown in Listing 1.

Listing 1: Opening the application

```

on preOpenStack
    set the defaultmenubar to the name of group "MainMenu"
    
```

```

set the defaultcursor to arrow
if the QTVersion is "0.0" then
    answer error "QuickTime is not installed!"
    close stack "RunRevVeez"
end if
end preOpenStack
    
```

Notice that a message handler is delimited by the keywords "on" and "end", each followed by the name of the message. The `preOpenStack` message is sent to a card before the card is drawn in its stack on the screen. In our case, we want to perform any one-time initialization of the mainstack and the application. We set the default menu bar and the default cursor, and we check to see whether QuickTime is installed. The built-in function `QTVersion` returns the version of QuickTime that is installed on the system. If QuickTime is not installed, then `QTVersion` returns the value "0.0". As you can see, we look for that value; if we find it, we use the `answer` command to display the alert box shown in Figure 7. (The `error` keyword indicates that we want the error icon to be displayed; other choices are `information`, `question`, and `warning`.)

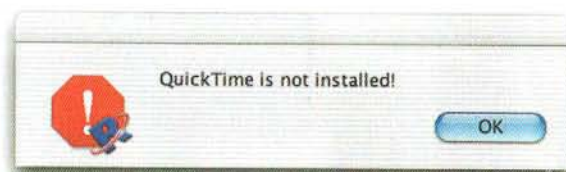


Figure 7: Hiding the controller bar

If QuickTime is not available, we quit `RunRevVeez` by closing the mainstack, like so:

```
close stack "RunRevVeez"
```

In general we refer to stacks and objects by name.

Setting Up the Menus

Let's move next to set up our application's menus and menu bar. With Revolution, this is a bit trickier than with other RAD tools, owing largely to the fact that Revolution wants to target a variety of operating systems. Menus in Revolution are buttons, and the menu bar is a group of buttons. Each stack can have its own group of buttons — that is, its own menu bar. On Macintosh operating systems, this isn't really what we want. So we'll create a single menu bar associated with the mainstack and then set the application's `defaultmenubar` property so that all the stacks in our application use that menu bar. (Look again at Listing 1.)

To create our menus, select "Menu Builder" in the Tools menu. The *menu builder* dialog box shown in Figure 8 appears.

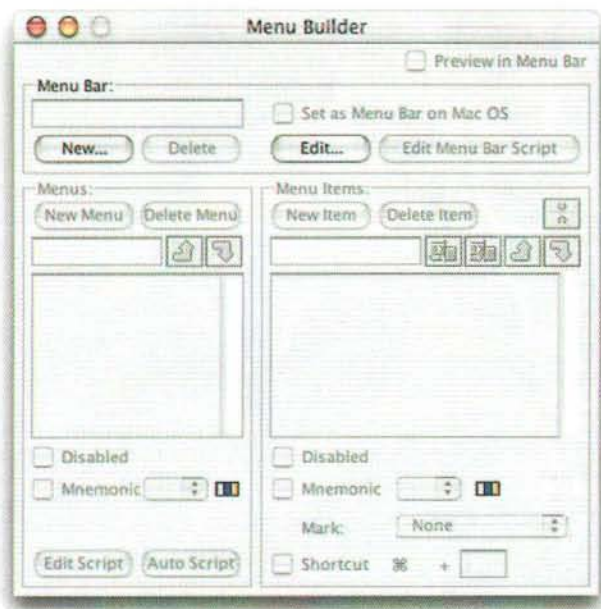


Figure 8: The menu builder dialog box (empty)

Click the New button. In the dialog box that then appears (Figure 9), set the name to "MainMenu". Notice that three menus are already included in the new menu bar.

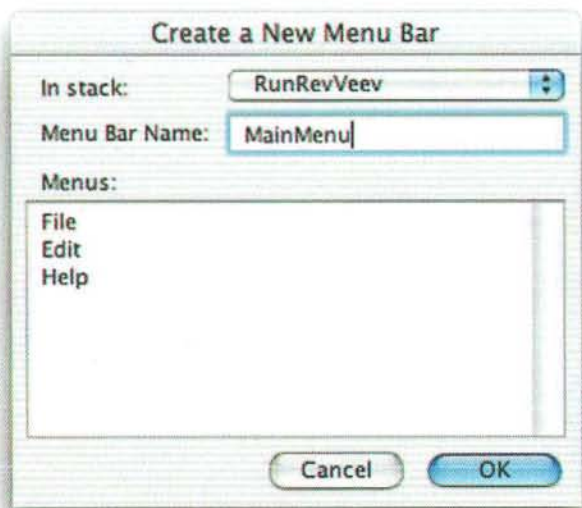
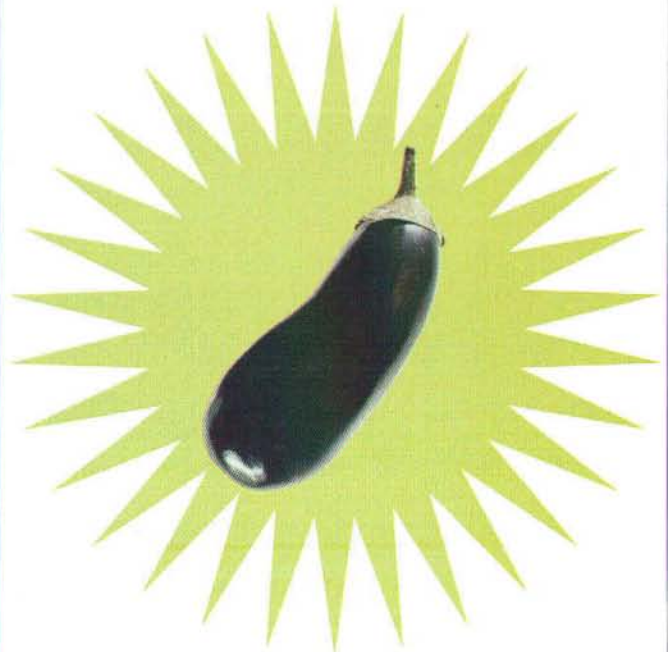


Figure 9: The menu naming dialog box

When we click the OK button, the menu builder dialog box now contains the three default menus. Click the "Set as Menu Bar on Mac OS" check box (as in Figure 10) to get the proper behavior on Macintosh systems.

The newest addition
to every **GREAT**
developer's diet.



Easy to learn and use
Complete more testing faster
Test any application on any system
Allows testing of remote systems
Captures test results visually

eggplantTM
Test any application on any system.

www.macegg.com

Redstone
SOFTWARE

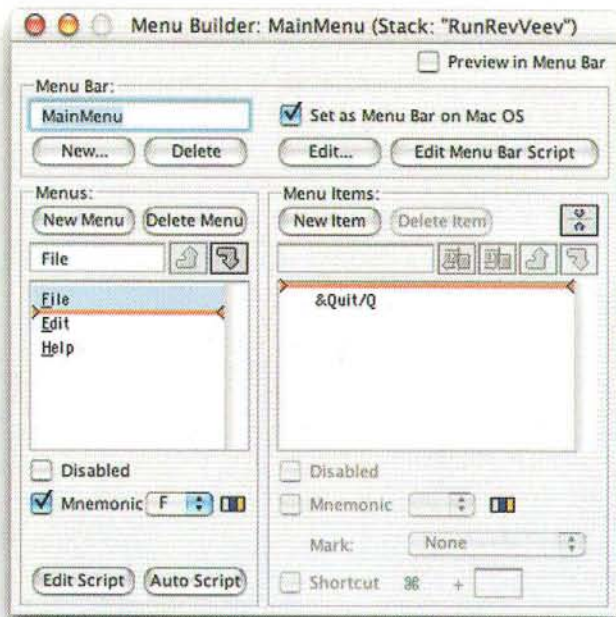


Figure 10: The menu builder dialog box (default)

The construction of menus is reasonably straightforward. Using the controls in the menu builder dialog box, add the appropriate menu items (together with the usual keyboard shortcuts) to the File and Edit menus. Add a new menu, the Movie menu, and add two items to that menu: "Hide Controller Bar" and "Hide Speaker Button". In the Help menu, add the item "About RunRevVeez". When we're done, the menu builder dialog box will look something like Figure 11.

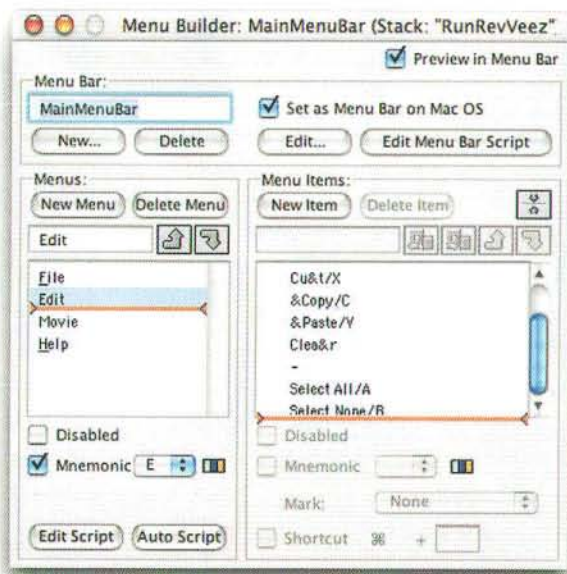


Figure 11: The menu builder dialog box (final)

Now we need to add some code to handle the various menu items. In turn, select each of the four menus in the menu builder dialog box (File, Edit, Movie, and Help) and click the "Auto Script" button. Each time we click that button, Revolution creates a default script for the selected menu. For instance, Figure 12 shows the default script for the Help menu.

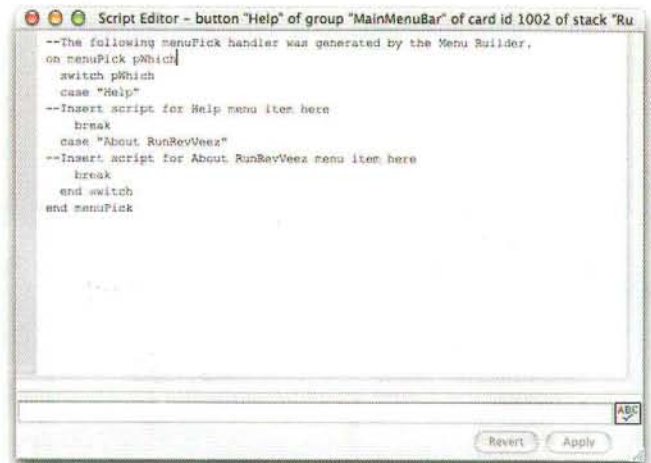


Figure 12: The Help menu script (default)

Notice that the Menu Builder has created a `menuPick` message handler, replete with the necessary `switch` and `case` statements to handle all the items we added to the Help menu. All we need to do is add code to handle those items in the appropriate manner. In RunRevVeez, we'll simply ignore the Help item in the Help menu. To handle the "About RunRevVeez" menu item, we can insert the code in Listing 2 into the `menuPick` handler.

Listing 2: Handling the About menu item

```

menuPick
case "About RunRevVeez"
    open stack "AboutBox"
    set visible of stack "AboutBox" to true
    break
end switch

```

This tells our application to open a stack named "AboutBox" and then to set its `visible` property to true.

It turns out that Revolution is smart enough to move the "About RunRevVeez" menu item from the Help menu into its proper place in the Mac OS application menu, as shown in Figure 13.

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."

— MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."

— MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."

— Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"

— Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."

— Bill Goodman, author of *Smaller Installer* and *Compact Pro*

"Resorcerer Rocks! Buy it, you will NOT regret it."

— Joe Zobkiw, author of *A Fragment of Your Imagination*

"Resorcerer will pay for itself many times over in saved time and effort."

— MacUser review

"The template that disassembles PICT's is awesome!"

— Bill Steinberg, author of *Pyro!* and *PBTools*

"Resorcerer proved indispensable in its own creation!"

— Doug McKenna, author of *Resorcerer*



Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater,
1.5MB RAM, CD-ROM

Standard price: \$256 (decimal)
Website price: \$128 - \$256
(Educational, quantity, or
other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMAESTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New
in
2.0:

- Very fast, HFS browser for viewing file tree of all volumes
- Extensibility for new Resorcerer Apprentices (CFM plug-ins)
- New AppleScript Dictionary ('aete') Apprentice Editor
- MacOS 8 Appearance Manager-savvy Control Editor
- PowerPlant text traits and menu command support
- Complete AIFF sound file disassembly template
- Big-, little-, and even mixed-endian template parsing
- Auto-backup during file saves; folder attribute editing
- Ships with PowerPC native, fat, and 68K versions

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com



Figure 13: The Application menu of RunRevVeez

Adding the About Box

Of course, our application does not yet contain a stack named "AboutBox". So let's add one. Select the menu item "New Substack of RunRevVeez" in the IDE's File menu. A new stack (that is, window) will appear. Open the property inspector for the new stack and set its name to "AboutBox" and its title to "About RunRevVeez".

Next, click the "magic wand" icon next to the Controls field; a sheet will slide down allowing us to select the desired controls for the window's title bar, as shown in Figure 14.

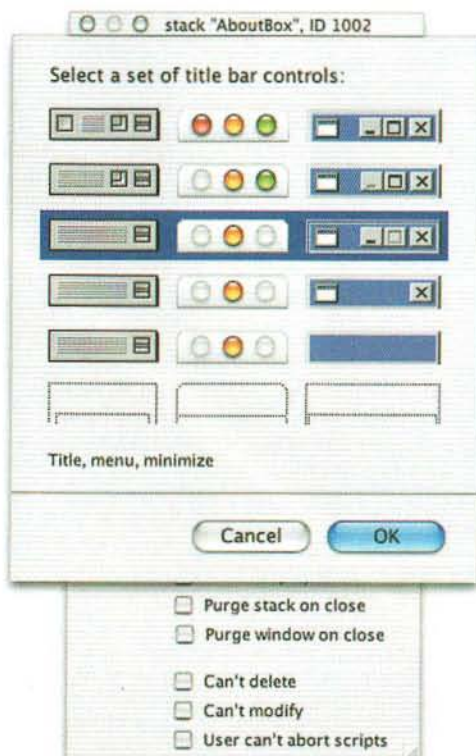


Figure 14: The window title bar control sheet

Let's select the set of controls with the fewest buttons; since we are currently building RunRevVeez only for Mac OS, we can select the third set from the top. Click OK.

Select the "Colors & Patterns" item in the pop-up menu in the properties inspector for the About box and then choose the Aqua striping for the background pattern. Then select the "Size & Position" item and set the size of the stack as shown in Figure 15.

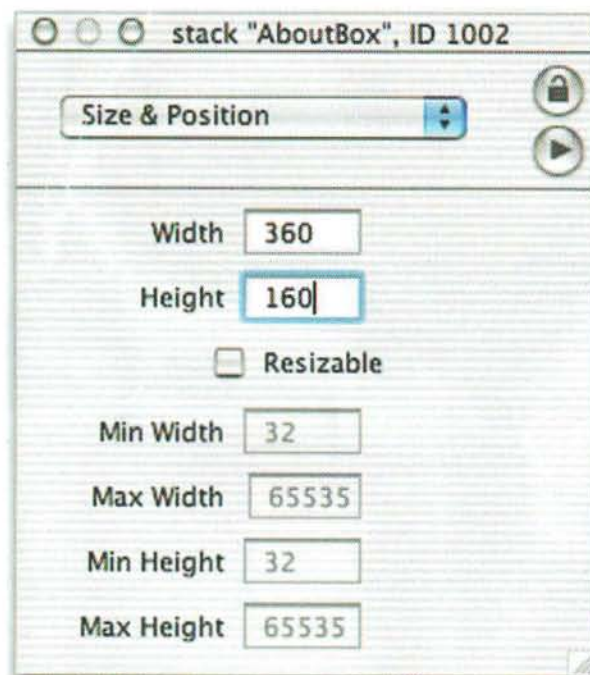


Figure 15: The About box size settings

Now we need to add items to the About box so that it has the standard appearance shown in Figure 16. (Notice that I've added a one-pixel border around the image; as we'll see, this is easy to do, and it makes the box look a bit nicer.)

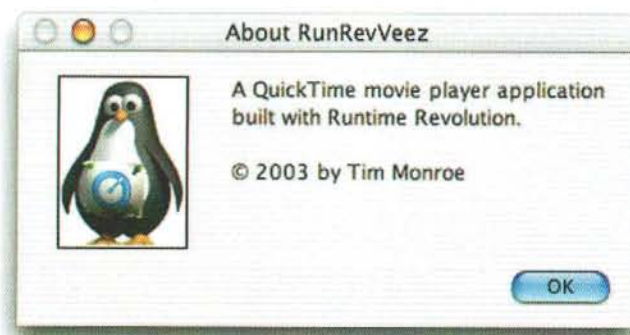


Figure 16: The About box of RunRevVeez

First, click the Image tool in the tools palette and then click-drag an image well in the About box stack. Set the size of the image as shown in **Figure 17**.

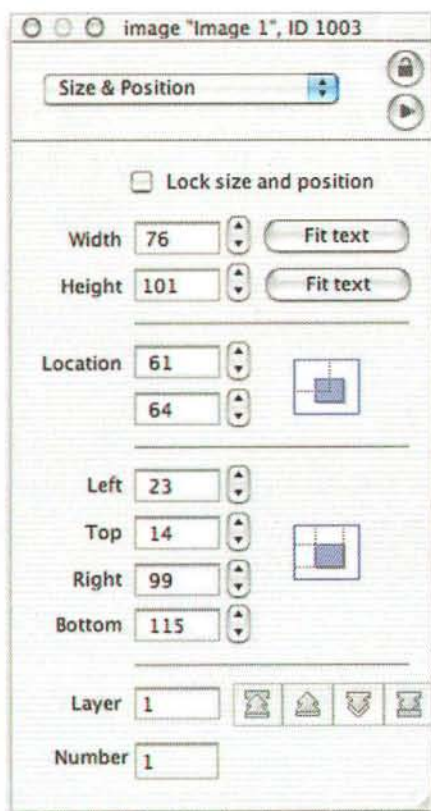


Figure 17: The penguin image size settings

Set the image file to the standard penguin image, `penguin.pct`. In the properties inspector for the image, turn on the "Show Border" property but turn off the "Three D" option for that property. Finally, set the border width to 1. Set the tool tip for the penguin image to "Keep the lawyers happy!"

Apparently, the Revolution license requires any applications built using it to display the Revolution logo at some point. (Hey, I'm not a lawyer, so this might not be strictly correct.) So let's add a second image to the About box; this image will be initially invisible but will become visible when we click the penguin image. Add a second image using one of the logos distributed with the Revolution IDE, say "`Runtime small.pct`". Make that image initially invisible. Open the script editor for the penguin image and add the handler shown in **Listing 3**.

Listing 3: Handling clicks on the penguin image

```
on mouseUp
    set the visible of me to false
    set the visible of image "RevIcon" to true
end mouseUp
```

SOFTWARE LOCALIZATION MADE

easy

POWERGLOT FEATURE HIGHLIGHTS

- ▶ LOCALIZE CLASSIC, CARBON™, COCOA® AND PALM OS™ APPS
- ▶ LEVERAGE EXISTING TRANSLATIONS
- ▶ AUTOMATE WITH APPLESCRIPT®
- ▶ IMPORT /EXPORT TRANSLATION MEMORIES

www.powerglot.com
browse, translate, click, done.

PowerGlot Software • Localization tools for Mac® OS
www.powerglot.com
info@powerglot.com

Mac OS, Carbon, Cocoa and AppleScript are trademarks of Apple Computer, Inc.
Palm OS is a trademark of Palm, Inc.

Once the user has clicked the penguin image, the About box will look like the box in **Figure 18**.

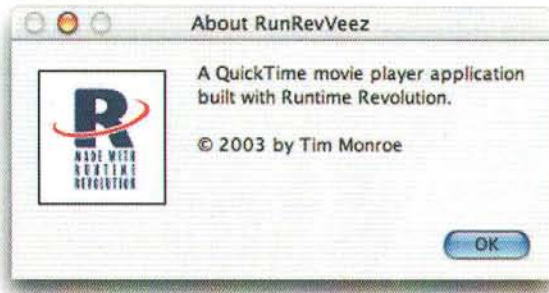


Figure 18: The About box with the Revolution logo

Just for fun, let's configure the Revolution icon image so that clicking it will turn it back into the penguin image. Of course the script would look like the code in **Listing 4**.

Listing 4: Handling clicks on the revolution logo image

```
on mouseUp
    set the visible of me to false
    set the visible of image "Penguin" to true
end mouseUp
```

The last control we need to add to our About box is the OK button. Select the button icon in the tools palette and then click-drag a button of the appropriate size in the appropriate spot. In the properties inspector, set the name of the button to OKButton and the label to OK. Check the "Default button" check box. Finally, we need to script the button so that clicking it causes the About box to disappear. This is easy enough: add a `mouseUp` message handler to the button with this single line of code:

```
close this stack
```

Our About box is now complete. It's time to move on to constructing a movie window.

REVOLUTION'S PLAYER OBJECT

Revolution provides a *player object* (or, more briefly, a *player*) that we can use to display QuickTime movies in a card. There are a large number of built-in messages and properties associated with a player object, and we can also define custom messages and properties. Of particular interest is the `movieControllerID` property, which is a pointer to the movie controller associated with the movie being displayed in a player object. As we'll see in more detail in the next article, we can use this property to extend Revolution's QuickTime capabilities beyond the built-in methods and properties.

These built-in properties include a number of properties associated with normal linear QuickTime movies:

`currentTime`, `timeScale`, `startTime`, `endTime`, `paused`, `playLoudness`, `playRate`, `playSelection`, `trackCount`, `looping`, and so forth. For QuickTime VR movies, Revolution defines a handful of properties, including: `currentNode`, `hotspots`, `nodes`, `pan`, `tilt`, `constraints`.

Most of these properties are associated with numerical or Boolean values, but several of them return a *list* of values. Revolution provides easy ways to iterate through these lists looking for particular values. For instance, a player's `mediaTypes` property contains a comma-separated list of strings that describe the media types in a movie; this list can contain some or all of these words: `video`, `audio`, `text`, `qtvr`, `sprite`, `flash`. We can find movies with sound tracks, and operate upon them, like this:

```
if the mediaTypes of player "MoviePlayer" contains "audio"
then
    incrementLoudness
end if
```

Revolution also defines eight messages that can be sent to a player object: `currentTimeChanged`, `deletePlayer`, `hotspotClicked`, `newPlayer`, `nodeChanged`, `playPaused`, `QTDebugStr`, and `selectionChanged`. For present purposes, however, we won't need to write any message handlers for these messages.

Adding a Movie Window

Let's begin by adding yet another substack to the mainstack. This new substack will serve as a template for new movie windows. When the user decides to open a movie file, we'll clone this stack and load the movie into the cloned stack. In this way, we'll have the ability to open an unlimited number of movie windows, all sharing the properties of this substack.

Name the new substack "MovieWindow". Then add a player object to it by clicking on the QuickTime icon in the tools palette and then click-dragging in the new substack. (If you prefer to use menus, you can select the Player item in the "New Control" hierarchical menu of the Object menu.) Don't worry about making the player exactly fill the movie window, as we'll take care of resizing the movie window when we load a movie into it. In the properties inspector for the player object, set the player's name to "MoviePlayer". Make sure that the Visible, Controller, and "Focus with keyboard" check boxes are checked. More importantly, make sure that the Buffer check box is unchecked. (Checking that box would turn on the `alwaysBuffer` property of the player object, which renders the controller bar ineffective.) **Figure 19** shows the desired properties.

Xserve



Density optimized rack mounted Mac OS X Server



UNIX-based Server Solutions from Apple
Nearly half a terabyte of storage per 1U
630 gigaflops of processing power
Hot-Swappable drives
Industry-standard 1U rack-optimized design

There has never been a better time to buy...
Small Dog Electronics carries
factory refurbished models too
offered at substantial savings!
(subject to availability)



**Small Dog
Electronics**
www.smalldog.com

1673 MAIN STREET, ROUTE 100, WAITSFIELD, VERMONT



Apple Specialist

To learn more: <http://www.smalldog.com/xserve/>

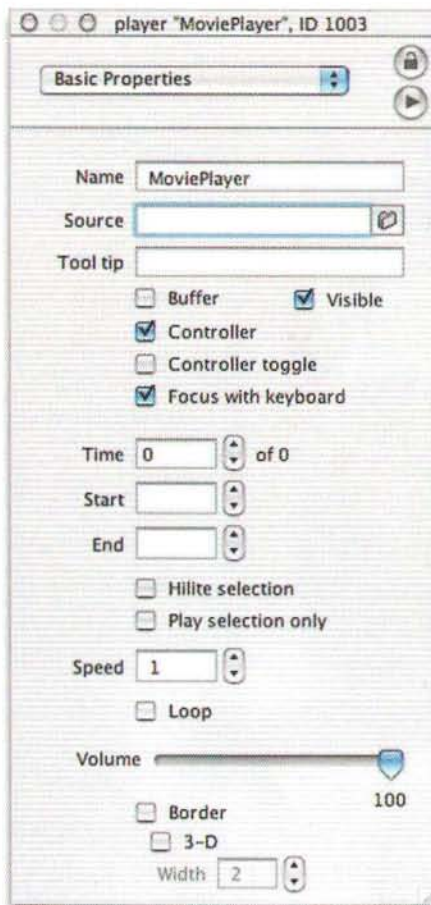


Figure 19: The player object properties

Notice two things here. First, the Border option is not selected; since the movie window will be sized to exactly contain the movie and its controller bar, we don't want a border around the movie. Second, notice that we've left the Source property blank. If we wanted to open a specific movie, we could specify its full pathname as the source. As usual, however, we want to give the user complete choice over which movies to open. So we'll need to specify the source movie programmatically in our script.

Opening a Movie File

Now let's see how to open a movie file and display the movie it contains in a movie window. Obviously, we want to add some code to the menuPick handler for the Open menu item in the File menu. Revolution provides the "answer file" command that we can use to elicit a movie file from the user; we can call it like this:

```
answer file "Open: RunRevVeez" of type "MooVSWFL"
```

This line of code displays the standard file-opening dialog box, with the prompt string (here "Open: RunRevVeez") used as the

title of the dialog box; in this case, only movie files and Flash files can be selected by the user.

When the user selects a file, the complete pathname of that file is placed into a special variable named "it". We can make sure that the user actually selected a file (and didn't just hit the Cancel button) by testing that it isn't the empty string:

```
if it <> "" then
```

And we can save that full pathname for later use by assigning it to a local variable:

```
put it into savedFilePath
```

Recall that we are going to make a copy of the MovieWindow substack to serve as a window to hold the new movie. We can do that quite easily:

```
clone stack "MovieWindow"
```

The new window is named "Copy of MovieWindow", and this will also be the title of the new window if we don't change it. As usual, we want the title of the movie window to be the file's basename (the portion of the full pathname that follows the rightmost path separator). We can get the basename quite easily like this:

```
set the itemDelimiter to "/"
put the last item of savedFilePath into newStackName
```

The itemDelimiter is the character that is used to delimit the chunks of a string. By default, a handler's itemDelimiter property is set to the comma (","), but it can be reset at will. Here we set it to the slash character ("/"), which delimits the parts of a pathname. Once we've executed these two lines of script, the local variable newStackName should contain the file's basename. We can then use that variable to set the name of the stack and the title of the new movie window:

```
set the name of stack "Copy of MovieWindow" to newStackName
set the title of stack newStackName to newStackName
```

Now we need to assign the movie in the file we just opened to the player object in the stack we just created. We can do that by setting the filename property of the player object in that stack, like this:

```
set the filename of player "MoviePlayer" of stack
newStackName to savedFilePath
```

The filename property can be set to any full or partial pathname; if the pathname is partial, then it's interpreted relative to the value of the defaultFolder global property. (When a Revolution application is launched, the defaultFolder property is set to the folder containing the application; we can programmatically set it to any other folder we wish.) The filename property can also be set to a URL. In RunRevVeez,

however, we'll always use full pathnames, as returned by the "answer file" command.

Listing 5 shows our complete definition of the menuPick handler for the Open menu item.

Listing 5: Opening a movie file

```

case "Open..."
answer file "Open: RunRevVeez" of type "MooVSWFL"
if it <> "" then
    put it into savedFilePath

    -- get the base name of the file
    set the itemDelimiter to "/"
    put the last item of savedFilePath into newStackName

    -- create a new movie window
    clone stack "MovieWindow"
    set the name of stack "Copy of MovieWindow" to
        newStackName
    set the title of stack newStackName to newStackName
    set the visible of stack newStackName to false

    -- load the movie into the new window
    set the filename of player "MoviePlayer" of stack
        newStackName to savedFilePath

    -- set the size of the movie window
    sizeStackToMovie newStackName
    set the visible of stack newStackName to true

    set the visible of player "MoviePlayer" of stack
        newStackName to true

end if
break

```

You'll notice that we make the movie window invisible before setting its size and then make it visible thereafter. We also need to make sure the player object itself has its visibility state restored.

Sizing Movie Windows

The last thing we need to do when opening a QuickTime movie is to resize the movie window so that it exactly contains the movie at its natural size and the movie controller bar, if present. In **Listing 5**, we called the message handler `sizeStackToMovie`, which is defined in the mainstack. The definition is straightforward and relies on the height and width properties of the player object (**Listing 6**).

Listing 6: Setting the size of a movie window

```

on sizeStackToMovie theStack
    put the height of player "MoviePlayer" of stack theStack
        into windowHeight
    put the width of player "MoviePlayer" of stack theStack
        into windowWidth

    set the width of stack theStack to windowWidth
    set the height of stack theStack to windowHeight
    set the rectangle of player "MoviePlayer" of stack
        theStack to 0,0,windowWidth,windowHeight
end sizeStackToMovie

```

We set the height and width of the movie window to the height and width of the player object, which is set to the movie's natural dimensions when the movie is first opened. Then we

There are Users

and Losers...

Which
would you rather
be

DATA
Backup

DATA
Recycler

DATA
Rescue

Data Users get it!

www.prosofteng.com
PROSOFT
engineering, inc.

These products are only available for the Mac OS, so your
Windows friends will still be losers...

©2003 Prosoft Engineering, Inc., All rights reserved

reset the rectangle of the player object so that its upper left corner is at the point 0,0 in the window's content area.

MOVIE PLAYBACK

The amount of code we need to write in order to open a QuickTime movie in a window on the screen is surprisingly small. We clone the `MovieWindow` stack, set its `filename` property to a path elicited from the user, and then resize the movie window to exactly contain the movie at its natural size. Once we've done this, the movie appears, just like it should, in a window; **Figure 20** shows a sample movie window displayed by our current version of `RunRevVeez`.

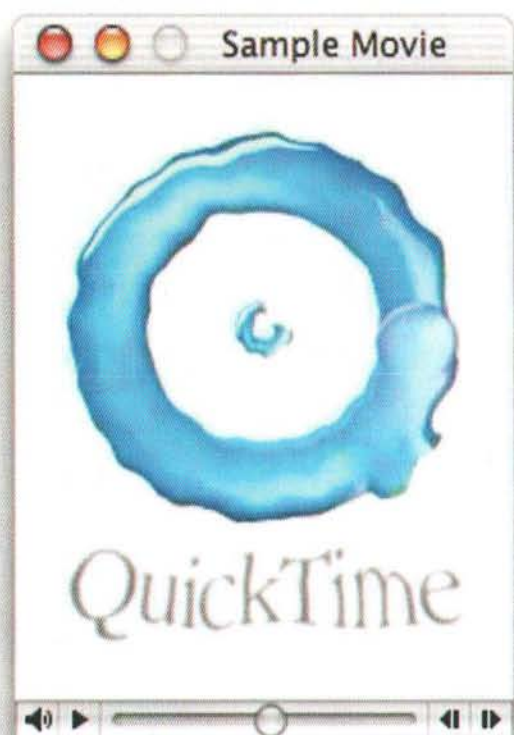


Figure 20: A sample movie window

This window behaves exactly as we'd expect; we can drag it around on the screen by grabbing its title bar, minimize it by clicking on the yellow button in the title bar, and close it by clicking on the red button in the title bar. We configured the window so that it's not resizable, so the green button is inactive.

Moreover, the controller bar functions perfectly, and keyboard operations (such as starting and stopping the movie by pressing the space bar, or moving forward and backward by pressing the right and left arrow keys) work exactly as they should. In short, for normal linear QuickTime movies and for QuickTime VR movies, Revolution works flawlessly as far as normal movie playback is concerned.

Ensuring Smooth Playback

Problems arise, however, when we open certain types of non-linear QuickTime movies, like movies containing wired sprites or zero-source QuickTime video effects. For instance, when we open the fire movie we created in an earlier article ("F/X" in *MacTech*, September 2001), we'll quickly notice that the fire does not burn by itself. (See **Figure 21**.) A few flames sputter up initially and then the fire freezes (if that's even possible). Oddly enough, if we open a linear movie and start it playing, the fire will join in and burn, but only for as long as the linear movie is playing. Even more oddly, if we open the About box, the fire starts burning, but (again) only for as long as the About box is open.



Figure 21: A non-burning fire

It's possible that I've got my movie stacks somehow configured incorrectly, but I'm inclined to doubt that, given that we've used such simple and natural code to get to this point. It seems more likely to me that the Revolution runtime engine is for some reason failing to call `MCIsPlayerEvent` or `MCIdle` often enough to keep these non-linear movies playing. I dunno.

Happily, the About box behavior described above suggests a simple and apparently effective workaround. Keep in mind that our mainstack is currently empty, visually speaking. It's serving only to hold handlers for application start-up and window sizing. I have discovered that if we place a default button on the mainstack, as shown in **Figure 22**, then the fire movie and wired sprite movies behave as they should. Something about that pulsating button seems to wake up the movie controllers. I dunno.

Ask yourself this question...

I use my Mac for:

- ☐ Programming as a hobby
- ☐ Exploring the depths of Mac OS X
- ☐ QuickTime Development
- ☐ Application Development
- ☐ Network Administration
- ☐ All Of The Above

...this is the answer:

 **MacTech[®]**

The Journal of Macintosh Technology and Development

Whether you program as a hobby or are a full-time developer, MacTech gives you the info you need from the people that know.

www.mactech.com

PO Box 5200 • Westlake Village, CA • 91359-5200 • orders@mactech.com
Toll-Free: 877-MACTECH • Outside US/Canada: 805-494-9797 • Fax: 805-494-9798



Figure 22: The main stack with a button

Of course we don't want the mainstack to be visible while our application is running, especially with its throbbing button. So let's just move it out of the way at application launch time. In the `preOpenStack` message handler we wrote earlier (see **Listing 1**), let's add one line at the bottom:

```
set the location of stack "RunRevVeez" to 5000,5000
```

This moves the stack sufficiently far offscreen that it shouldn't be visible.

As I said, this workaround seems to solve the problem entirely. There may be a better solution to the original problem, but this hidden-mainstack-with-throbbing-button will do fine until we find that better solution.

Handling the Movie Menu

Recall that we added a couple of menu items to the Movie menu, one of which is supposed to toggle the visibility state of the controller bar. We can implement this item quite easily in Revolution by getting and setting the player object's `showController` property, as shown in **Listing 7**.

Listing 7: Setting the size of a movie window

```
case "Hide Controller Bar"                                menuPick
  if showController of player "MoviePlayer" of stack
    theTopStack is true then
    set showController of player "MoviePlayer" of stack
      theTopStack to false
  else
    set showController of player "MoviePlayer" of stack
      theTopStack to true
  end if
  sizeStackToMovie theTopStack
break
```

Notice that we call `sizeStackToMovie` to make sure that the movie window is resized correctly to hold the movie and any associated controller bar.

Listing 7 raises an interesting question: how did we figure out the value of `theTopStack`? `RunRevVeez` can have more than one movie window open at a time, and we want to apply the menu item only to the frontmost movie window. So how do we figure out which movie window is in front of all others? Revolution provides the `topStack` function, which returns the frontmost stack window with the lowest mode. Since the mainstack and the movie window stack have the same mode, `topStack` ought to return the frontmost movie window.

The trouble is, I could never get `topStack` to work quite right. Fortunately, Revolution also provides the `openStacks` function, which returns a list of open stacks ordered according to their order on the screen. The first element in this list is the frontmost stack, and we can read that first line like this:

```
put first line of the openStacks into theTopStack
```

You can expect to see that line of code at the beginning of any handlers for menu items that apply only to a movie window. You will also often see the following line of code, which tests to see whether the frontmost window contains a player object:

```
if exists(player "MoviePlayer" of stack theTopStack) then
```

This allows us to distinguish a movie stack from the mainstack.

CONCLUSION

So far, so good. Revolution appears to be a simple yet powerful environment for creating applications. It provides support for opening and displaying QuickTime movies, and it's easy enough to get our application to support multiple open movies at once. We ran into a minor problem with playback of some kinds of movies, but we also discovered a simple and effective workaround to that problem.

Still, we've got a fair amount of work remaining in our quest to develop a multi-window movie playback and editing application. We need to tackle the movie editing operations, and we need a way to track the state of our movie windows so that we can enable and disable menus and menu items appropriately. For these enhancements, we need to move beyond what's built in to Revolution and explore the topic of Revolution plug-ins. We'll tackle all that in our next article.

CREDITS

Thanks are due to Kevin Miller and Tuviah Snyder at Runtime Revolution Ltd. for their unflagging assistance in improving my understanding of Revolution. A very special thanks is due to Geoff Canyon of Inspired Logic, LLC, for answering a number of questions about Revolution.

The Runtime Revolution web site (<http://www.runrev.com>) contains a wealth of information about Revolution as well as a free trial version of the development environment.



Sound Screen

The Original Sound Conditioner



only
\$49.95

Day or night. Home or office!

This compact device produces a gentle whooshing noise that helps block out intermittent or continuous annoying sounds such as traffic and ticking clocks so that you can relax and fall asleep easily. It can also be used during the day to mask distracting sounds for anyone requiring a quiet environment for concentration.

Sound is fully adjustable for tone and volume and has two frequency levels for broader range. High-impact plastic case. Rocker switch for soft and amplified settings. Plugs into household outlet. 3-1/2" H x 6-5/8" Diam.

www.radgad.com

Toll Free: 877-5-RADGAD (877-572-3423) • Outside US/Canada: 805-494-9797 • Fax: 805-494-9798

By Matt Campbell

Ch - A C/C++ Interpreter

New possibilities for people who like C and Unix.

FIRST LOOK

Given that C is the basis for much of the programming behind the Mac, which now utilizes Unix as its substructure, C seems to be the language that all Macs speak. If you're not eager to dive into many different languages for different tasks, take a look at Ch. Ch expands the capabilities of C and makes it an interpreted language. Its tight integration with the Unix shell makes it natural for Mac OS X. To those who are using C and are comfortable with the command line interface or are willing to learn, I think you will discover Ch to be a useful tool. Ch can help with shell programming, development, numerical computing, and learning C.

Setup and Trying It Out

Anyone wanting to give it a whirl can get the Standard version of Ch for OS X, Unix, and Linux free. Not bad for starters. Check out the website at <http://www.softintegration.com/>. Go to the downloads page and register, after which you will be sent download instructions via email (mine came in a matter of seconds). After downloading and installing the package, start the Terminal application in the Applications > Utilities folder and type `ch -d`. The `-d` option loads the startup file, `.chrc`, in your home directory. Then edit this file using `vi` and uncomment the line that starts with `_path`. This lets you run files on the command line without entering the current directory path, `'./'`, before the file name. There's a little information in the README file including directions to uninstall. If you like, Ch can even be run as your default shell. Using the Terminal > Preferences... menu, select the button to execute this command upon creating a terminal window and enter `/bin/ch`. Now any new shell you start will automatically be running Ch. More documentation can be found in the `/usr/local/ch/docs` folder from the command line by typing `open filename.pdf`. I've been using Ch for 6 months and C covers lots of ground, so I'll try to give you the run down on most of the important features of Ch. Here it goes.

CH AS A C/C++ INTERPRETER

At the simplest level, Ch is a C interpreter. It is able to run C code without compiling. It does not need a project space or any of the other associated files that are usually created in the compile/build/debug/execute cycle by most programs used in writing code. All that's necessary is the program file. There are two levels in its ability to run the C language on the fly. First, it will run C code interactively from the command line, as a command interpreter. If you're still learning C, this is invaluable. When having problems with a particular statement or wondering about the output of a command, it can be entered on the command line to test how it works. This works not only for simple assignment statements, but loops and conditional statements as well. It is sort of "programming as you go" when coupled with the Terminal's drag and drop ability. You can test several lines of code and if they work properly, just drop them in your program. Second, programs can be executed from the command line without compiling, with Ch acting as a program interpreter. Many people like this option for testing out small pieces of code or being able to run programs in stages, testing new functions as you build them into the program. The process of the compile/debug cycle can be skipped altogether. This allows for fast changes to a program and quick concept testing. A nice feature of Ch comes up when running programs this way. It's error messages are not the cryptic type usually given by a compiler when a program has problems. When the program fails, Ch prints out the offending line and line number and also points to the suspected error with arrows. Most of the time, I find this feature speeds the process of correcting small errors and oversights since you don't have to pick through a lot of lines to find where the problems are hiding. As an example of interpretive C, here's the classic "hello world" program run in four ways. First is the actual program:

```
#include <stdio.h>
int main()
{
    printf("\nhello world.\n\n");
}
```

Matt Campbell is a Graduate Student in Mechanical Engineering at University of California, Davis. He has become a devoted Mac user over the past 2 years and has recently entered the world of programming. He hopes that more engineering apps will be developed for the Mac so he can finally convince his colleges that Macs are the way to go. Feel free to provide some feedback at mtycampbell@ucdavis.edu.

```
Matty's Terminal — ch — ch
Ch
Professional edition, version 3.7.8.11231
(C) Copyright 2001-2003 SoftIntegration, Inc.
http://www.softintegration.com
/Users/matty> cd ch/ch review/programs
/Users/matty/ch/Ch Review/programs> hello.c

hello world.

/Users/matty/ch/Ch Review/programs>
```

Figure 1. Ch shell running *hello.c* as an Interpreter

Figure 1 above shows the first method, the program being run by typing the name at the command prompt. Ch uses '>' as the default prompt. Along with the program file, you just need any associated header files and library files if you programmed them separately. Below are the other ways. In the second, the command is interpreted directly and executed. Third, the same thing is done with C++ code. More about that in the next section. The fourth way is the simplest. All that is needed is quotes and Ch interprets this as a string.

Interpreting C code on the command line:

```
> printf("hello world.")
hello world.
```

C++ interpreted:

```
> cout << "hello world." << endl
hello world.
```

Interpreting a string directly:

```
> "hello world"
hello world
```

This is another example of code executed on the command line. Here I actually defined a simple function and then used it as you normally would inside a program.

```
> void input(void){char a[20],b[20];scanf("%s",a);
scanf("%s",b);printf("\n%s\n",streat(a,b));}
> input()
Mac
Tech
MacTech
```

Your ONE-STOP Price Comparison for Movies!

MOVIE DEPOT sm

www.moviedepot.com

In addition, the two methods of interpreting C can be combined. Programs can be run from the command line with `. program.c` (that's a period followed by a space then the program name) in the current shell. The functions used in the program will remain loaded in memory after the program is run. Then they can be used interactively on the command line. This works great for more complex functions that you'd like to try out. Class definitions and all the related member functions for C++ code work the same way. All the member functions can be tested interactively. By running an empty program that includes any necessary header files, all the functions can be tested without having to write a program that uses them. I like being able to try out functions this way to help me visualize program flow. The `parse` shell command is similar. Used for debugging, `parse program.c` checks for syntax errors and keeps functions loaded in memory without running the program. Variables have to be declared in the shell; drag and drop works nicely here. Once created though, they can be used in function calls just as in a program. Return values from functions can even be passed back and assigned to variables in the shell.

Ch is not just for simple programs; it can handle larger ones. When studying Dynamics recently, I used a program called `Autolev` that symbolically formulated the equations of motion of a system of bodies and then wrote a C program that numerically solved the resulting differential equations. In my particular case, I sought to simulate the simplified kicking motion of a person. `Autolev` wrote a program that came in at over a thousand lines, including the recursive numerical functions. Ch ran the program without any problems and without the need to go to a compiler. The best part was that with only slight modifications, I was able to get visual output in the form of graphs by adding just a little bit of code. I'll have an example of graphing in Ch a little later. This was great because not only was the feedback immediate, but it also kept me from having to import my data from multiple output files into a spreadsheet every time I ran the simulation.

SUPPORTING C90/C99/C++ STANDARDS

The ability to run C interpretively is nice. Add to this the fact that Ch is built to completely encompass the C90 standard, the standard that most C code is based on, and you have a good place to develop code. Ch goes beyond this, adding functionality to C by bringing in some of the newest standards, borrowing ideas from other programming languages, and blending in new functions to enhance C. Ch adds in many useful features from the new standard for C, C99. In keeping up with the times, Ch supports **complex** and **long long** type variables, floating point arithmetic, variable length arrays, polymorphism for generic functions, binary type output for `printf` statements, and the `//` comment

style. Most of the new standards that are supported in Ch are those that help in writing code for numerical computing. Along this same line, Ch has added some practical features from C++. It supports classes with public and private members, reference type and pass-by-reference, and the `cout`, `cin`, and `cerr` functions to name some of the main ones. Thus, object oriented programming is possible without having to go completely to the C++ language. As C++ does not necessarily support all the newest features of the C99 standard, Ch becomes a mixture of the two, combining some features of both. The following is a simple program that uses classes. Immediately after it, you can see where I used the `. classes.cpp` syntax, both running the program and loading the classes into memory. I then used the class definition and functions on the command line.

```
// test uses of class
#include <iostream.h>

class Triangle
{
public:
    Triangle(double, double);
    void hypotenuse();
private:
    double m, n, h;
};

int main()
{
    class Triangle t1 = Triangle(3, 4);
    t1.hypotenuse();
    class Triangle t2 = Triangle(5, 12);
    t2.hypotenuse();
    return 0;
}

Triangle::Triangle(double a, double b)
{
    m = a;
    n = b;
}

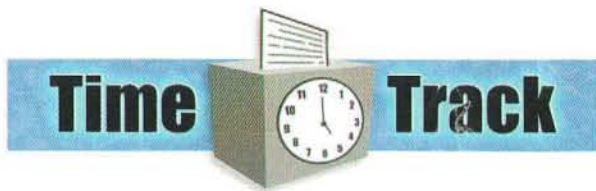
void Triangle::hypotenuse()
{
    h = sqrt((m*m) + (n*n));
    cout << "Geometry of a Right Triangle" << endl;
    cout << "Sides = " << m << " and " << n << endl;
    cout << "Hypotenuse = " << h << "\n" << endl;
}

// End of file

> . classes.cpp
Geometry of a Right Triangle
Sides = 3.0000 and 4.0000
Hypotenuse = 5.0000

Geometry of a Right Triangle
Sides = 5.0000 and 12.0000
Hypotenuse = 13.0000

> class Triangle t1 = Triangle(2, 3)
> t1.hypotenuse()
Geometry of a Right Triangle
Sides = 2.0000 and 3.0000
Hypotenuse = 3.6056
```



Keep track of every second of your time and bill for it with Time Track! Built in instructions make it easy to use. It is a simple way to manage your billable time for multiple projects and create a web page to show to your clients. Only \$24.95 per single user license per platform. Finally! A versatile time tracking solution for Macintosh, Windows, and Palm!

www.trinfinitysoftware.com



By TLA Systems

The Dock with more than one dimension.

Create multiple docks of any size, assign hot keys and even put the Trash back on the Desktop. A flexible and feature-laden tool for power-users. Runs natively on Mac OS X and 9 in five languages.

"...you made the switch to OS X a lot easier for me..." - Bob LeVitus

"...DragThing can rightfully be called an indispensable aid to working with your Mac..." - MacUser UK

Download a copy now from www.dragthing.com.



Trapcode - plug-ins for Adobe® After Effects®

Trapcode Shine is a fast light effect plug-in. The effect looks very much like volumetric light, but is actually a 2D effect. There are special controls to make shimmering lights and numerous coloring modes. This is an effect that you see everyday on TV and in many movie titles. Shine is available for Mac, Mac OSX and Windows.

www.trapcode.com



Eudora Internet Mail Server (EIMS) 3.2 is the latest version of the most popular Internet mail server for the Macintosh. If you need to handle email for a dozen users, or thousands of users, EIMS is a reliable and easy to use solution.

EIMS 3.2 is available for US\$400.00, there are no limits on the number of users that can be added, and free email support is included.

For more information, see

<http://www.eudora.co.nz/>



GraphicConverter converts pictures to different formats. Also it contains many useful features for picture manipulation.

See www.lemkesoft.com
for more information.

String t type

Ch also comes with new data types and functions. Strings have been promoted to a first class object using a new data type, `string_t`. This type automatically handles memory allocation as needed, frees used memory when appropriate, and allows strings to be used in symbolic computing. Strings can also be used to control the number of iterations in a loop. A `foreach` loop has been added, which breaks a string into tokens and runs through the loop once for each token, as demonstrated below. This function has a man page within Ch.

```
/* Using the foreach loop */
#include <stdio.h>

int main()
{
    string_t loop, token;
    loop = "this string has been broken up";
    foreach(token; loop)
        printf("%s\n", token);
    return 0;
}
// End of file

> loop.c
this
string
has
been
broken
up
```

As with strings, arrays can be used in new ways as well. Assumed shape arrays, `name[:][:]`, are allowed in function calls where the shape of the array is not determined until run time. Array subscripts can be declared explicitly, meaning that the first element of an array can be element 1 or even -1, instead of 0.

Computational Array Type

Computational arrays have been added as a new data type, `array type name[m][n]`, which allows them to be treated as a first class object. These can be thought of as matrices and vectors for use in calculations. This is where the numerical part of Ch really shines. In C, arrays have to be handled with many levels of embedded loops. In Ch, conversely, the solution to a matrix equation or printing an array can often be done with one line. Output statements can have matrix equations as arguments and print them as if they were single numbers. Matrix math can be done as simply as floating-point math. `Array_1 * Array_2` will perform the matrix multiplication according to the rules of Linear Algebra. The `.*` operator multiplies corresponding elements of two arrays times one another. Generic math functions also work with the new data type. The following is an example of computational array type, how it can be passed to functions, and using it for computations and output without loops.

```
/* matrix calculations in Ch */
#include <stdio.h>
```

```
#include <array.h>

array double function(array double a[:][:],
                      array double b[:][:], int c[:][:]);

int main()
{
    array double A1[3][2] = {1, 2, 3, 4, 5, 6};
    array double A2[2][2] = {1, 2, 4, 5};
    array double B1[4][2] = {1, 3, 5, 7, 10, 11, 12, 13};
    array double B2[2][2] = {7, 8, 3, 6};
    int C1 = 1, C2 = 5;

    printf("\nPart 1:\n%3g", function(A1, B1, C1));
    printf("\nPart 2:\n%4g", function(A2, B2, C2));
    return 0;
}

array double function(array double a[:][:],
                      array double b[:][:], int c[:][:])
{
    return c * (a .* a) * transpose(b);
}
// End of file

> matrix.c

Part 1:
 13 33 54 64
 57 157 266 316
133 377 646 768

Part 2:
 195 135
1560 990
```

Here, similar arrays and functions are interpreted on the command line:

```
> array int x[6]
> array int y[6]
> linspace(x,1,6)
6
> x
1 2 3 4 5 6

> y
0 0 0 0 0 0

> void add(array int *p, array int *q){q=p+p;}
> void square(array int *s, array int *t)
    {t=s.*s;printf("s=\n%d\tt=\n%d\n",s,t);}
> add(x,y)

> y
2 4 6 8 10 12

> square(x,y)
s=
1 2 3 4 5 6
t=
1 4 9 16 25 36

> y
1 4 9 16 25 36
```

New functions have been added in Ch to aid in using computational arrays, most of which are for numerical computing. This allows for complex computations to be done in C without going to another math program. Unfortunately, Ch does not include the ability to do symbolic manipulations, nor does it have the depth of numerical functions that some

```
> shape(b)
20
```

professional level math programs have, but the most important computations can be handled numerically. Statistics, data analysis, Fourier transforms, vector algebra, integration, and differentiation are just a few of the computational algorithms that have been included. They are available in the header file `numeric.h`. Most of the matrix manipulations used in Linear Algebra, such as transpose, inverse, or more complicated ones, can be done as demonstrated in the example above.

Using computational arrays, an array of reference can be passed to a function, which can handle multiple data types and different dimensions. One function can be given much more flexibility this way. Functions can also return computational arrays of varying dimensions, making it easy to give values back to the calling function. The `shape()` function is a simple way to get array dimensions when using an array of reference since its size is not passed. Shown here with both computational and C arrays, it returns the dimensions of the argument. By using it twice, you can get the total number of dimensions.

```
> array double a[5][6]
> shape(a)
5 6

> shape(shape(a))
2

> double b[20]
```

TWO & THREE DIMENSIONAL PLOTTING

As a compliment to the increased numerical capabilities, Ch has also added high level plotting through function calls that look and feel like C. Both 2-D and 3-D plots can be implemented with simple function calls, or through a class designed to handle plotting named **CPlot**. The plotting functions, e.g. `plotxyz()`, will accept all the default display options. The only arguments needed are two or three arrays, either computational or C arrays, with the correct number of points. You can add axis labels and a title with optional inputs. These functions are great for getting quick feedback by adding one line of extra code to a program. By using the **CPlot** class, many more options are available through the member functions. Data can be added in sets to allow for multiple curves or surfaces. Subplots can be generated that occupy the same window but use different data sets or are different types of plots. Plotting can be done to the screen or directed to a file with a specified type. I have found PNG files to be the most useful as they are read by Preview, but you can also output a postscript file if you use an application that will read them. There's quite a bit of flexibility given the relatively simple commands that are used. Plots can be rectangular, polar,

Who has the best fixed wireless solution in the industry?



There is no comparison!

Effective Data Throughput ⁽¹⁾ <small>(Not to be confused with Signaling Rate)</small>	9.5 Mbps	Dynamic Bandwidth Control ⁽²⁾	Yes
Range ⁽²⁾	Fade margin: SU, SU-EXT 6 dB, 7 mi, 20 mi 12 dB, 3.5 mi, 10 mi	Latency Control	Yes
Users/AP	500	RF Thresholding ⁽⁴⁾	Yes
Non-Overlapping Channels	6	User Interface	Telnet, HTTP, Serial
Software Controlled Antenna Polarization	Yes	Price - SU/AP ⁽⁵⁾	\$495/\$895

(1) Measured on SmartBits 600 platform with 1518 byte sized packets.
(2) Trango specifies ranges with a 12 dB fade margin for added robustness.

(3) Indicates the ability to dynamically change up/down stream data throughput for unmatched bandwidth efficiency.
(4) A feature to minimize RF interference not found in competing systems.
(5) Low volume; ask us about higher volume discounts.

Trango Broadband is committed to providing the best technology at the most affordable price for your broadband wireless access deployment. Check us out and see why our Access5800™ solution is the best in class. Give us a call...Don't settle for less anymore!

trangobroadband
WIRELESS
A division of Trango Systems, Inc.

Call Now!

858-653-3900

EMAIL: sales@trangobroadband.com
www.trangobroadband.com

FIXED WIRELESS...THAT WORKS!



(Ask about our third party leasing program)

cylindrical, or spherical. 2-D plots can be simple lines, steps, impulses, or points. 3-D surfaces can be drawn as a mesh, impulses, or points all with or without contours. The view orientation for 3-D plots can be changed. There are many more options available for graphs and demos of each can be found on the Softintegration website. Below, I've created two simple examples of plots done in Ch. **Figure 2** was done on the command line.

```
> array double x[100]
> array double y[100]
> linspace(x, 0, 2*M_PI)
100
> y = sin(x)
0.0000 0.0634 0.1266 0.1893 0.2511 0.3120
0.3717 0.4298 0.4862 0.5406 0.5929
0.6428 0.6901 0.7346 0.7761 0.8146 0.8497
0.8815 0.9096 0.9341 0.9549 0.9718
0.9848 0.9938 0.9989 0.9999 0.9969 0.9898
0.9788 0.9638 0.9450 0.9224 0.8960
0.8660 0.8326 0.7958 0.7557 0.7127 0.6668
0.6182 0.5671 0.5137 0.4582 0.4009
0.3420 0.2817 0.2203 0.1580 0.0951 0.0317
-0.0317 -0.0951 -0.1580 -0.2203 -0.2817
-0.3420 -0.4009 -0.4582 -0.5137 -0.5671
-0.6182 -0.6668 -0.7127 -0.7557 -0.7958
-0.8326 -0.8660 -0.8960 -0.9224 -0.9450
-0.9638 -0.9788 -0.9898 -0.9969 -0.9999
-0.9989 -0.9938 -0.9848 -0.9718 -0.9549
-0.9341 -0.9096 -0.8815 -0.8497 -0.8146
-0.7761 -0.7346 -0.6901 -0.6428 -0.5929
-0.5406 -0.4862 -0.4298 -0.3717 -0.3120
-0.2511 -0.1893 -0.1266 -0.0634 0.0000

> plotxy(x, y, "Sin(x)", "X", "Y")
0
```

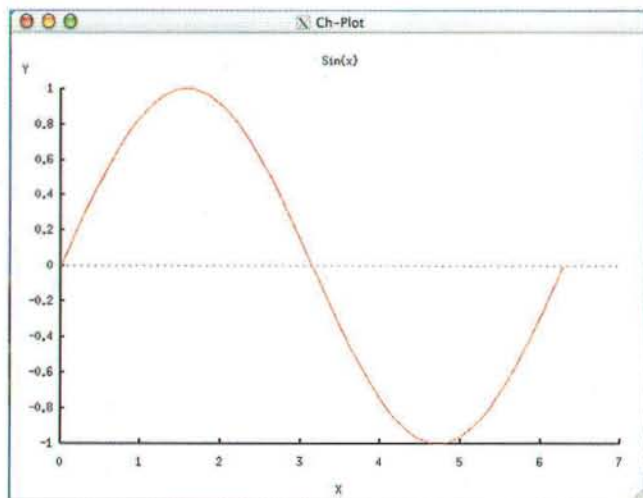


Figure 2. Plotting Sine wave from command line

Figure 3 below shows the output of the plot.c program. Here plotting is done with the **CPlot** class.

```
/* plot.c */
/* 3D Plotting */
#include <stdio.h>
#include <math.h>
#include <chplot.h>
```

```
# define NUMPTS 30

int main()
{
    //Variables
    array double x[NUMPTS], y[NUMPTS], z1[NUMPTS*NUMPTS],
        z2[NUMPTS*NUMPTS];
    array double levels[6];
    int i, j;
    string_t xlabel = "x", ylabel = "y", zlabel = "z",
        title = "3D Surfaces";
    class CPlot plot;

    // Create data for x, y, and levels arrays
    linspace(x, -1, 1);
    linspace(y, -1, 1);
    linspace(levels, -1, 1);

    // Computation of 2 surfaces
    for(i = 0; i < NUMPTS; i++)
    {
        for(j = 0; j < NUMPTS; j++)
        {
            z1[NUMPTS*i+j] = x[i]*x[i] + y[j]*y[j] - 1.0;
            z2[NUMPTS*i+j] = sin(6*x[i]) * sin(6*y[j]);
        }
    }

    // Enter arrays to class "plot"
    plot.data3D(x, y, z1);
    plot.data3D(x, y, z2);
    // Set options for graphing
    plot.changeViewAngle(60, 60);
    plot.contourMode(PLOT_CONTOUR_BASE);
    plot.contourLevels(levels);
    plot.title(title);
    plot.label(PLOT_AXIS_X, xlabel);
    plot.label(PLOT_AXIS_Y, ylabel);
    plot.label(PLOT_AXIS_Z, zlabel);
    // Graph results
    plot.plotting();

    // Create output files for printing
    plot.outputType(PLOT_OUTPUTTYPE_FILE, "png color",
        "plot_file.png");
    plot.plotting();
    return 0;
}
```

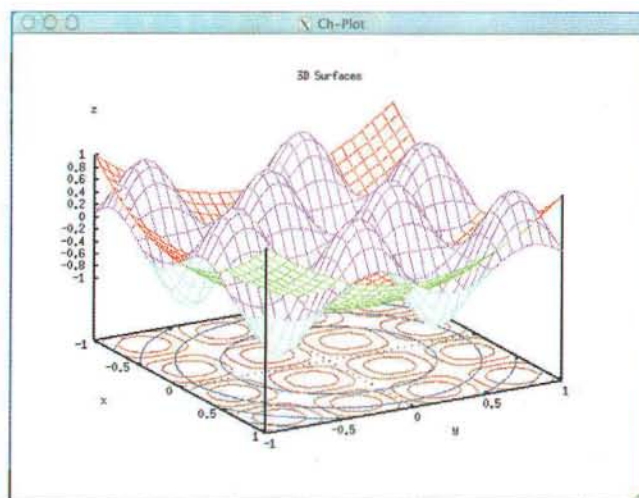


Figure 3. Two 3D surfaces from program plot.c

COMMAND SHELL & SHELL PROGRAMMING

Perhaps the most useful feature of Ch is its ability to run as a shell. You interface with Ch on the command line from within Terminal or the X11 application. It runs just like a Unix shell with all the same commands available. It can be started from within any of the shells that come with the Mac, be it bash, sh, csh, tcsh, or zsh. You start it by typing `ch` at a command prompt. You can also start these other shells from within Ch, if you so desire. While this may mean no fancy interface with lots of buttons or menus, it's good news for the dedicated Unix users who like the fact that they can now wrap their hands around Unix on the Mac. To people unfamiliar with a Unix terminal, as I was a year ago, this is an opportunity to learn. I mean, who wouldn't want to know how to use the vi editor? You may not end up using the terminal as your primary interface for day to day work since the Aqua interface is much more human friendly, but no doubt you'll find it useful with the mass of information that's out there on the topic of everything Unix. If you end up using Ch regularly while running and testing code, you can make Ch the default shell used by Terminal, as mentioned in the Setup section. No need to run the default tcsh shell or any other shell at all. You can also save a Terminal file that runs Ch. Just set Ch as the default shell, open a new one, and set all your other preferences; then use Terminal's Save As command to store this shell for quick access from the Library menu.

The fact that Ch runs as a Unix shell has another significant meaning. While Ch will run scripts the same as any other shell, the fact that Ch is a C interpreter gives you have the ability to include C code in your scripts. Loops and conditional statements can use the more familiar C syntax. If the first line of the shell program is `#!/bin/ch`, you don't even need to be running Ch to be able to execute the script with included C code. To compliment this, scripts can be called and run directly from inside of a program. No special include function is necessary, just the script name and path, if it happens to be in a different folder. These features will give anyone more options when it comes to shell programming. Not having to learn another syntax for writing scripts is nice too. The script below, saved as `scripting.ch`, uses C right along side some regular shell programming.

```
#!/bin/ch
#include <stdio.h>

echo Sample script
int i;
for(i = 0; i < 3; i++)
{
    printf("hello %d\n", i);
}

int transfer(void);

echo Making copy of file
cp ./test1.txt ./test3.txt
```

```
echo Copying contents
transfer();

echo moving new file
mkdir ./testfolder
mv ./test3.txt ./testfolder/test3.txt
echo done!
```

```
int transfer(void)
{
    FILE *in, *out;
    char c;

    if(!(in = fopen("test2.txt", "r")))
    {
        perror("_tmpfile");
        return -1;
    }
    in = fopen("test2.txt", "r");
    out = fopen("test3.txt", "a");
    while((c = fgetc(in)) != EOF)
    {
        fputc(c, out);
    };
    echo Closing files
    fclose(in);
    fclose(out);
    return 0;
}
```

// End of file

```
> scripting
Sample script
hello 0
hello 1
hello 2
Making copy of file
Copying contents
Closing files
moving new file
done!
```

Ch adds an additional feature to make running scripts and programs even simpler. If the file is named with the extension `.ch`, it is not necessary to include the extension on the command line to run the script. It's automatically recognized as shown in the above example. At the prompt following the script, just typing the filename without the extension is enough to run it. As Ch runs programs interpretively, this works the same when running C programs too. The drawback here is that if you want to compile a program using a C or C++ compiler you will get an error unless you change the extension to `.c` or `.cpp`.

EASY INTERFACE WITH EXISTING C LIBRARY

Another nice feature coming out of the fact that Ch supports the current C standards is that you can use existing C libraries without modification under Ch. All of the existing tools that you've developed previously will not need to be rewritten as if you are transitioning to a new type of language. By utilizing the SDK package for Ch, even precompiled binaries can be included within Ch as in C.

POSITIVE AND NEGATIVE

So far I tried to cover all the features and positive things about Ch that I've run across in my personal experience using the program. To summarize, I like its tight fit with the

Mac and OS X. Considering that the Ch interface is basically a shell and that the Mac now comes with utilities to provide access to Unix, they seem to compliment one another. I believe the power of Ch's interpretive nature will aid anyone in shell programming and writing C code. Built on the standards of C and expanding on them to bring more potential to the language ensures that it will always be a useful tool. Don't forget that the Standard version of Ch is free for Mac OS X (Linux and Unix too). This alone will provide anyone with a simple, clean interface for running C and C++ code. This version doesn't include the computational array type, added numeric functions, or plotting, but it's a fantastic way to learn C interactively. Just be prepared to learn some shell commands if you've never used a terminal interface before.

That being said, there are a few areas where Ch could stand some improvement. One of the first things you might notice is the lack of a *Tab Complete* feature at the command line interface. This is a handy thing to have when navigating the command line. In Ch, long file names, which I am a fan of, have to be typed out completely. Dragging and dropping files from the Finder to the Terminal window can save you some of this work because this includes the entire path. Unlike Unix, thankfully, Ch is not case sensitive and can handle spaces without quotes or backslashes when it comes to file and folders names. This makes it a little easier to navigate through the folders that might give you longer names when using Aqua. Still, *Tab Complete* is sorely missed when in a Ch shell. Another issue crops up with file permissions. When a new file is created with vi or another text editor, it is not set to be an executable file so Ch will not run it. This might be an inherent feature of Unix for security reasons. You need to modify your file with `chmod 755 filename.c`. This will make it executable to everyone. Ch will then run the file as I have in the above examples.

There is a lot of documentation that comes with Ch in the form of PDF files located in `/usr/local/ch/docs`. The **Ch Users Guide** is almost 700 pages and covers not just Ch but much of the C language in general and many Unix commands. Although very thorough, it's a plain, utilitarian document so, as with most manuals, it is not always easy to read. It does have a lot of useful information such as the summary of the most common shell and vi commands and comparisons between Ch and other languages. The **Ch Reference Guide** is close to 1000 pages and gives detailed information on many C and Ch functions found in the most common header files. Those with quite a bit of programming experience can probably sift through these documents and pull out the stuff they need quickly. Beginners might want a different book to show them the basics of C.

Perhaps the biggest disappointment with Ch for the Mac is the lack of any kind of copy/paste ability when it comes to graphing. As shown above, graphs are drawn inside new windows in X11. You can't copy or drag the graph from here.

Screenshots seem impractical unless you use a slick screen capture program in lieu of Grab. I typically save graphs as PNG files from within the program and open them in Preview. This works well given all of Preview's export options, but it is still frustrating not being able to simply drag and drop the graph right from the window onto a document or copy and open it directly in Preview from the pasteboard. As this is a limitation of Unix and not Ch, it will not be changing anytime soon. It will only be another reminder of why we like the Mac interface so much. On the topic of graphing, Ch needs X windows in order to display graphs. The Ch website lists Xfree86 (XDarwin) with the OroborOSX window manager as system requirements. I have been using Apple's X11 (Beta 3) without any problems. The catch here is if you like using Terminal's more user friendly interface as opposed to the xterm in X11, you need to start the Terminal application from within X11, either through the Application menu or by modifying the `~/xinitrc` file. I actually learned how to do this in the "X11 Tweaks" article in the May 2003 issue of *MacTech*.

There's quite a bit of information here, but as this can't really cover it all effectively, I would urge anyone who's found this interesting to check out SoftIntegration's website. Aside from more information, example code is abundant. There is a Student version available and also a 30 day Demo for the Professional version. Scripting and numerical computing are definite strong points, but there are many other features to check out. Ch has to be the best and most intuitive way to learn the C language. Considering you can get a free simple tool to use the latest C standards interpretively, it's definitely worth a look.

BIBLIOGRAPHY AND REFERENCES

Cheng, Harry H. **The Ch Language Environment, User's Guide**. Revision 3.7. SoftIntegration, Inc. Davis, 2003.

Morin, Rich. "X11 Tweaks". **MacTech Magazine**. 19:5 (May 2003), pp. 12-15.

<http://www.softintegration.com/> - Homepage for Ch

POSTSCRIPT

Ch 4.0 was recently released. Starting with this version, Ch Standard Edition is freely available to everyone on all supported platforms. The Professional Edition is now free to all academic users for all platforms, giving them access to the computational and graphing features.

MacTech[®]
M A G A Z I N E

Get MacTech delivered to your door at a price **FAR BELOW**
the newsstand price. And, it's **RISK FREE!**

Subscribe Today!
www.mactech.com



Peachpit

Essential books for the creative community

Hot Off the Press!

Motivate your inner artist with these brand-new books on video editing, game designing, image editing, and more!

Developing Games in Java

By David Brackeen

1-59273-005-1 • \$49.99 • 1008 pages

Apple Pro Training Series:

Final Cut Express

By Diana Weynand

0-321-20039-X • \$44.99 • 528 pages

Secrets of the iPod, Third Edition

By Christopher Breen

0-321-22371-3 • \$19.99 • 328 pages

Adobe Acrobat 6.0 Standard Classroom in a Book

By Adobe Creative Team

0-321-19374-1 • \$45.00 • 456 pages

Editing Techniques with Final Cut Pro, 2nd Edition

By Michael Wohl

0-321-16887-9 • \$39.99 • 584 pages

Maya Character Creation: Modeling and Animation Controls

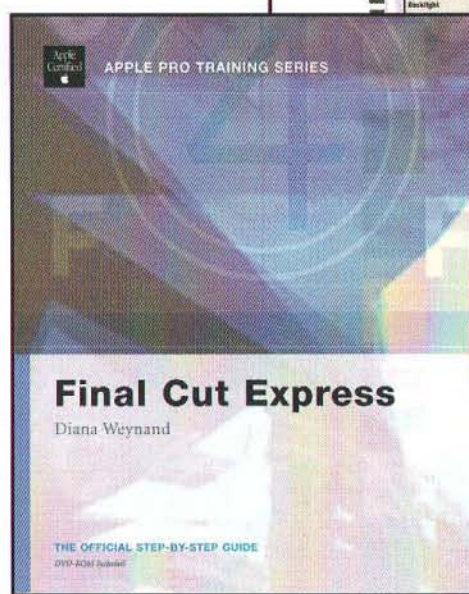
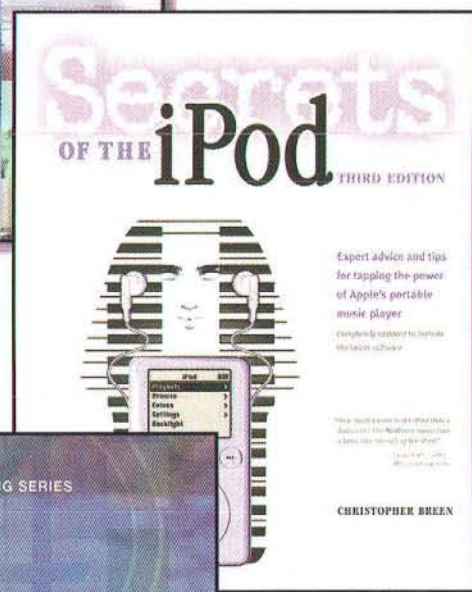
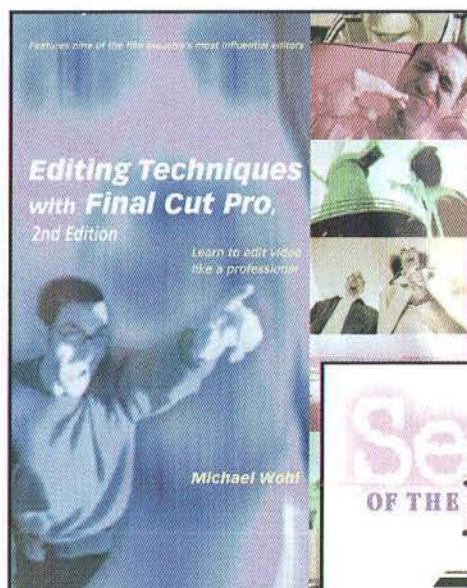
By Chris Maraffi

0-7357-1344-8 • \$45.00 • 408 pages

The Photoshop Show Starring Russell Brown

By Russell Brown

0-321-20042-X • \$35.00 • 304 pages



Save up to 30%! Become a Peachpit Club Member today!

Enjoy 10% off all books every day at peachpit.com, earn an additional 10% discount as a Peachpit Club Member, and save 10% on top of that with this one-time coupon! Simply go to www.peachpit.com/mactech903 and enter coupon code EM-93AA-MTMF at checkout. It's that easy!



Peachpit Press

New
Riders

By Vicki Brown

What's on Your Bookshelf?

Ten for X

BUILDING A LIBRARY

Macintosh users have, traditionally, prided themselves on the fact that they don't need to read manuals. While users of "those other operating systems" can't get by without "cracking the books", Mac users smile and flaunt the fact that the booklet that came in the box is still in its original shrink wrap. That's our traditional, public face.

In private, however, many Mac OS users actually do collect, read, write, and refer to books. We want to understand this operating system that we use daily. We want to know more about its intricacies and idiosyncrasies. We want to learn new tips and tricks that we can use to impress our friends. On the down side, Apple doesn't really supply a manual with Mac OS X. On the plus side, there are several publishers (and quite a few excellent writers) producing plenty of books on the subject.

GETTING STARTED

In 2002, the first edition of *Mac OS X: The Missing Manual* was the number one bestselling computer book. Created by David Pogue, computer columnist for The New York Times and bestselling Macintosh author, the "Missing Manual" series lays claim to providing the books "that should have been in the box" and lives up to its claim.

At 712 pages, *Mac OS X: The Missing Manual* is a sturdy tome, covering the Desktop, Finder, applications, components, the Unix underpinnings, and much more. Even Mac OS X power users should consider owning a copy of this book. At the very least, you'll want to recommend it to all of your friends and relatives who are still using Mac OS 9, as a gentle introduction to the wonders of Mac OS X.

A more advanced book, *Mac OS X Unleashed* (John Ray and William C. Ray), is aimed directly at power users and administrators. This book also starts by covering the basics, such as the Finder and applications; then it moves into the deeper administration arcana of networking, adding users, using the Terminal application, and the Unix command line. In fact, much of the book concentrates on capabilities that can be reached from the command line, rather than the Finder, and there are more textual examples than pictures throughout the book.

If you're a power user or administrator (or seek to become one), you'll want a copy of this book. My only gripe with it is that, at 1520 pages, it's more than a little awkward to handle!

If you're already at an advanced stage, but want a handy reference book, try *Mac OS X in a Nutshell* (Jason McIntosh, et. al.) Part of the O'Reilly Nutshell series, this book provides a more terse reference than the two mentioned above; Nutshell books "give you what you need to know in as few words as possible". This is the book you'll want to consult when you already understand the basics of what you need to do, but cannot quite recall the specifics. It also provides a handy way to review what you know, to make sure you haven't missed anything major. And, lest you worry that "terse" equates to less information – do not fear. At 800 pages, this book has plenty of "thud factor".

*NIX UNDER THE COVERS

Much has been made of the fact that Mac OS X is built on top of Darwin, a version of Unix based on BSD and the Mach microkernel. Although it is possible to use Mac OS X without ever making the acquaintance of the Darwin layer, power users, administrators, and developers (as well as Unix-users switching to Mac OS X!) will want to know more about Darwin. There are several books that provide assistance in this regard.

Two books from O'Reilly, *Learning Unix for Mac OS X* (Dave Taylor and Brian Jepson), and *Mac OS X for Unix Geeks* (Brian Jepson and Ernest E. Rothman), should certainly be on your shelf. Both are recommended by the Apple Developer Connection (ADC). *Learning Unix for Mac OS X* is aimed at Mac OS users who are interested in learning about Unix; *Mac OS X for Unix Geeks* is aimed at long-time Unix users who are "switching" to Mac OS X. Whichever group you see yourself in, I'd recommend adding both books to your shelf – if one doesn't address your questions, the other probably will.

If you want more depth than these two books provide, take another look at *Mac OS X Unleashed*. I was surprised at the amount of content this book provided on using the NetInfo Manager, using Terminal.app and basic Unix commands, and more. There are chapters on advanced shell concepts and commands, shell scripting, server and network administration, and much more. Consider this a heavy duty (in more ways than one :-)) book for advanced Mac OS X use.

TIPS, TRICKS, AND HACKS

Admit it; one of the coolest things about being a Mac OS X power user is the ability to amaze your friends and relatives with all kinds of nifty tips, shortcuts, and "how did you know that?" tricks. If you agree, you'll be happy to know that at least three books were written especially for you.

Mac OS X Killer Tips (Scott Kelby), deserves a place on your coffee table, if only for its look and feel. Every page is in full color, with screen shots. You'll find yourself flipping through this book simply because it's fun to do so and you'll learn new tricks at the same time.

Kelby says we're all drawn to the little "sidebar" tips in good books, because that's where the "really amazing, really fun, really useful stuff is found". So he decided to write a book that contained nothing but sidebar tips - 260 pages, 2 tips to a page. If you don't buy this book for the tips, buy it for the graphics; it's simply a fun book.

Mac OS X Hints, Jaguar Edition (Rob Griffiths) provides 560 tips on everything from the Desktop and Finder to Unicode, to popular third party applications. For those interested in the Unix command line, the book devotes two chapters (and 100 pages) to the terminal, shell, and other Unix hints.

A more technical set of tricks can be found in *Mac OS X Hacks* by Rael Dornfest and Kevin Hemenway. David Pogue, in the book's Foreword, writes:

This book might occasionally be over the head of many Mac fans. (If you want more general, less technical, everyday operating tips, try *Mac OS X Hints, Jaguar Edition*.)

But some people get as much a kick out of putting a computer through its paces as they do from everyday issues like productivity. Part of the spirit of hacking is doing things that the product's developer didn't quite imagine, finding the new and creative uses that are only possible to those who are willing to leave the beaten path. For the hackers among us, it's all about the thrill of discovery.

If that describes you, you'll want this book on your shelf.

UH OH...

Sometimes, despite all your precautions or learning, things go wrong. Ouch. Or, better, you may want to find ways to *prevent* things from going wrong. For those times, here are two more books you may want to have on hand.

Mac OS X Disaster Relief (Ted Landau) is the book you hope you'll never need. Landau is the founder of the MacFixit Web site (www.macfixit.com), the "primary first-aid station for Mac users in trouble". In this book, he makes his troubleshooting knowledge available to Mac OS X users, covering freezes and crashes (both prevention and recovery), third-party troubleshooting utilities, installing (and reinstalling) Mac OS X, printing problems, and more. If you have never had a

problem with Mac OS X, consider yourself blessed. However, you might still want a copy of this book on your shelf, as a good luck charm.

Mac OS X Maximum Security (John Ray and William C. Ray) provides "a hacker's guide to protecting your Mac OS X workstation and server". (Note: The media notwithstanding, hackers are the good guys; crackers are the bad guys.)

The original Mac OS (long before Apple called it "Mac OS") didn't need much security. As this book's authors put it, from a security standpoint, the early Macs might as well have been toaster ovens. They couldn't be compromised because, like toaster ovens, there was nothing to compromise. In recent years, however, the Mac OS has matured; the good news is that users have more and better features; the bad news is that security, once a no-brainer, has become an issue to consider.

Security is all about knowledge, risk assessment, and trust. The authors provide an introduction to the concepts and philosophy of computer security, including physical access, use policies, people, and limitations. The first nine chapters will give you a firm grounding in the theory and concepts of security (and how security can be compromised). Chapters 10 through 16 discuss specific Mac OS X resources and how to secure them, providing tips, tricks, and recipes. Chapters 17 through 20 concentrate on prevention, detection and reaction to attacks. If you're a system administrator, or you're hooking your Macintosh to the Internet, you may want to read this book... before you need it.

REFERENCES

- Learning Unix for Mac OS X*, 2e (Taylor & Jepson; O'Reilly; 2003; ISBN 0-596-00470-2)
- Mac OS X: Maximum Security* (Ray & Ray; Sams; 2003; ISBN 0-672-32381-8)
- Mac OS X: The Missing Manual*, 2e (Pogue; O'Reilly; 2002; ISBN 0-596-00450-8)
- Mac OS X Disaster Relief* (Landau; Peachpit; 2002; ISBN 0-201-78869-1)
- Mac OS X for Unix Geeks* (Jepson & Rothman; O'Reilly; 2002; ISBN 0-596-00356-0)
- Mac OS X Hacks* (Dornfest & Hemenway; O'Reilly; 2003; ISBN 0-596-00460-5)
- Mac OS X Hints: Jaguar Edition* (Griffiths & Pogue; O'Reilly; 2003; ISBN 0-596-004510-6)
- Mac OS X in a Nutshell* (McIntosh, et al; O'Reilly; 2003; ISBN 0-596-00370-6)
- Mac OS X Killer Tips* (Kelby; New Riders; 2003; ISBN 0-7357-1317-0)
- Mac OS X Unleashed*, 2e (Ray & Ray; Sams; 2003; ISBN 0-672-32465-2)

List of Advertisers

Aladdin Systems, Inc.	13
Big Nerd Ranch, Inc.	33
Circus Ponies Software, Inc.	11
DevDepot	28-29
EazyDraw (Dekorrra Optics, LLC)	43
effigent, Inc.	51
Electric Butterfly	10
Eudora Internet Mail Server	71
FairCom Corporation	1
Felt Tip Software	19
Fetch Softworks	9
Full Spectrum Software, Inc.	23
James Sentman Software	33
Lemke Software GmbH	71
Lingo Systems	47
MacDirectory	41
MacTech Magazine	65
Mathemaesthetics, Inc.	57
MCF Software	27
Morgan Kaufmann Publishers	39
Movie Depot	69
MYOB US, Inc.	21
/n software inc.	53
Netopia, Inc.	IFC
Paradigma Software	31
Peachpit Press	77
piDog Software	10
PowerGlot Software	59
Presto Vivace, Inc.	23
PrimeBase (SNAP Innovation)	37
Prosoft Engineering, Inc.	63
RadGad	67
Redstone Software, Inc.	55
Runtime Revolution Limited	82
Seapine Software, Inc.	15
Small Dog Electronics	61
Sophisticated Circuits, Inc.	35
Sophos, Inc.	7
Sybase, Inc.	2-3
The Iconfactory	20
ThinkFree Corporation	25
TLA Systems Ltd.	71
Trango Broadband Wireless	73
Trapcode Software	71
Trinifinity Software	71
Trolltech AS	81
Utilities4Less.com	45
VVI	17
WIBU-SYSTEMS AG	49

List of Products

Adobe Press • Peachpit Press	77
Big Nerd Ranch • Big Nerd Ranch, Inc.	33
Books • Morgan Kaufmann Publishers	39
c-tree Plus • FairCom Corporation	1
Development & Testing • Full Spectrum Software, Inc.	23
Disk Utilities • Prosoft Engineering, Inc.	63
DragThing • TLA Systems Ltd.	71
Eazy Draw • EazyDraw (Dekorrra Optics, LLC)	43
Eggplant • Redstone Software, Inc.	55
EIMS • Eudora Internet Mail Server	71
Enterprise Software • MCF Software	27
Fetch • Fetch Softworks	9
Graphic Converter • Lemke Software GmbH	71
InstallerMaker, StuffIt • Aladdin Systems, Inc.	13
IP*Works! • /n software inc.	53
Long Distance Phone Service • Utilities4Less.com	45
MacDirectory • MacDirectory	41
MacTech Magazine Subscription • MacTech Magazine	65
Maximizing Your Mac! • DevDepot	28-29
moviedepot.com • Movie Depot	69
MYOB • MYOB US, Inc.	21
Notebook • Circus Ponies Software, Inc.	11
piDog Utilities • piDog Software	10
PowerGlot • PowerGlot Software	59
PowerKey Pro & KickOff! • Sophisticated Circuits, Inc.	35
Presto Vivace Services • Presto Vivace, Inc.	23
PrimeBase • PrimeBase (SNAP Innovation)	37
Qt • Trolltech AS	81
Resorcerer • Mathemaesthetics, Inc.	57
Revolution • Runtime Revolution Limited	82
SmallIDog.com • Small Dog Electronics	61
Software Protection • WIBU-SYSTEMS AG	49
Sophos Anti-Virus • Sophos, Inc.	7
Sound Studio • Felt Tip Software	19
Stock Icons • The Iconfactory	20
Sybase services • effigent, Inc.	51
Sybase • Sybase, Inc.	2-3
TestTrack Pro • Seapine Software, Inc.	15
ThinkFree • ThinkFree Corporation	25
Timbuktu & netOctopus • Netopia, Inc.	IFC
Time Track • Trinifinity Software	71
Translation & Localization • Lingo Systems	47
Trapcode • Trapcode Software	71
UniHelp Module • Electric Butterfly	10
Useful Gifts & Gadgets • RadGad	67
Valentina • Paradigma Software	31
Visual-Report Tool Developer • VVI	17
WhistleBlower • James Sentman Software	33
Wireless Networking • Trango Broadband Wireless	73

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

UNFAIR



Qt/Mac, the Mac OS X version of the Qt multiplatform C++ application framework, provides a uniquely unfair technological advantage in software development. Qt/Mac lets Mac developers develop in C++ on the platform they love most, while targeting additional operating systems, such as Windows or Linux. Qt/Mac also offers:

- fully object-oriented, elegant API
- single-source, multiplatform development solution
- native compiling and applications that run at native speed



TROLLTECH™

- integration with Project Builder, allowing developers to use Apple's own IDE for editing, compiling and debugging Qt/Mac applications
- OpenGL support, for sophisticated 3D graphics on the Mac
- native look & feel, such as the Aqua style

Don't take our word for it. Visit us at www.trolltech.com/mac. Look at our list of Fortune 500 customers. Read what programmers say about Qt. Download the evaluation version, and see how easy C++ development can be.

Qt /MAC: THE UNFAIR ADVANTAGE IN C++ DEVELOPMENT

© 2003 Trolltech AS. All trademarks, registered marks and service marks are the property of their respective owners. All rights reserved.

It'll Give You A Life.

Nassau Beach, Bahamas - Sun, sand and soft breezes make this one of the most relaxing vacation spots in the world. Don't forget the cool, fruity drink!



Dear Revolution 2.0,

Thanks for the
great vacation!

Missing you terribly,

Love,

Cecil

Revolution 2.0
c/o My Computer 64.23.0.192
24-7 Relief From Aggravation
(formerly DullAndDreary Coders)
The Corporate World, #1-EASY

But A Geek Is Always A Geek.

**Revolution 2.0: the English like language designed around the way you think.
Develop and deliver on Mac OS X, Windows and Linux
(not to mention 10 other platforms).**

Take the English language, add elegant XML, more SQL databases than you care to learn, intrinsic video capture, Unicode, CGI scripting, sophisticated Reports, and build your solution faster than anyone else. Jaguar, Panther, XP? Revolution lets you eat platforms for breakfast. And speaking of eating, our Cookbook of examples gets you up, running, and productive NOW. You can join the Revolution for as little as \$99...
Build with it, deliver with it, love it - and still have time for vacations.

And for a limited time, your MacTech special lets you get 100% of the Revolution for 15% off - go to special.runrev.com NOW. Thousands of developers have already joined.
Don't let the Revolution in modern coding start without you...

Software At The Speed Of Thought



Software Development & Consultancy

Runtime Revolution • 91 Hanover Street • Edinburgh EH2 1DJ • UK
Phone +44 (0)131 718 4333 • Fax +44 (0)131 718 4334 • www.runrev.com • Email info@runrev.com